UCIe Standard Package Physical Layer

Abdallah Mohamed Salah

Abdelrahman Adwe Ali

Mohamed Elsayed Ebrahim

Momen Mostafa Mohamed

Mohamed Anwar Abd Allatif

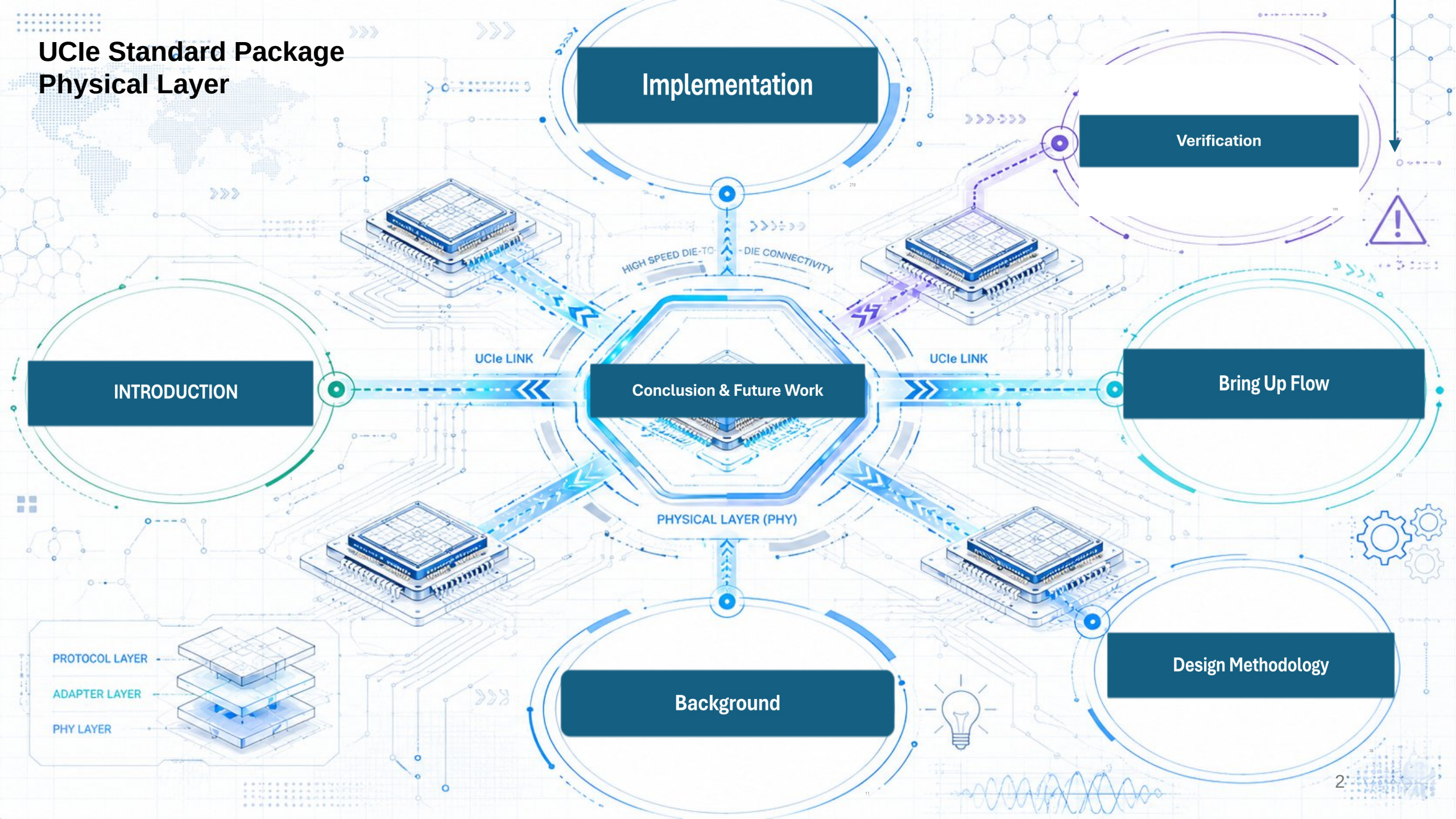
Yousef Mohamed Abdelfattah

Mohamed Adel Abdelaty

Sondoss Mohammad Hassan

Under supervision of :
Dr. Ali A. Al-Hamid

UCIe Standard Package Physical Layer



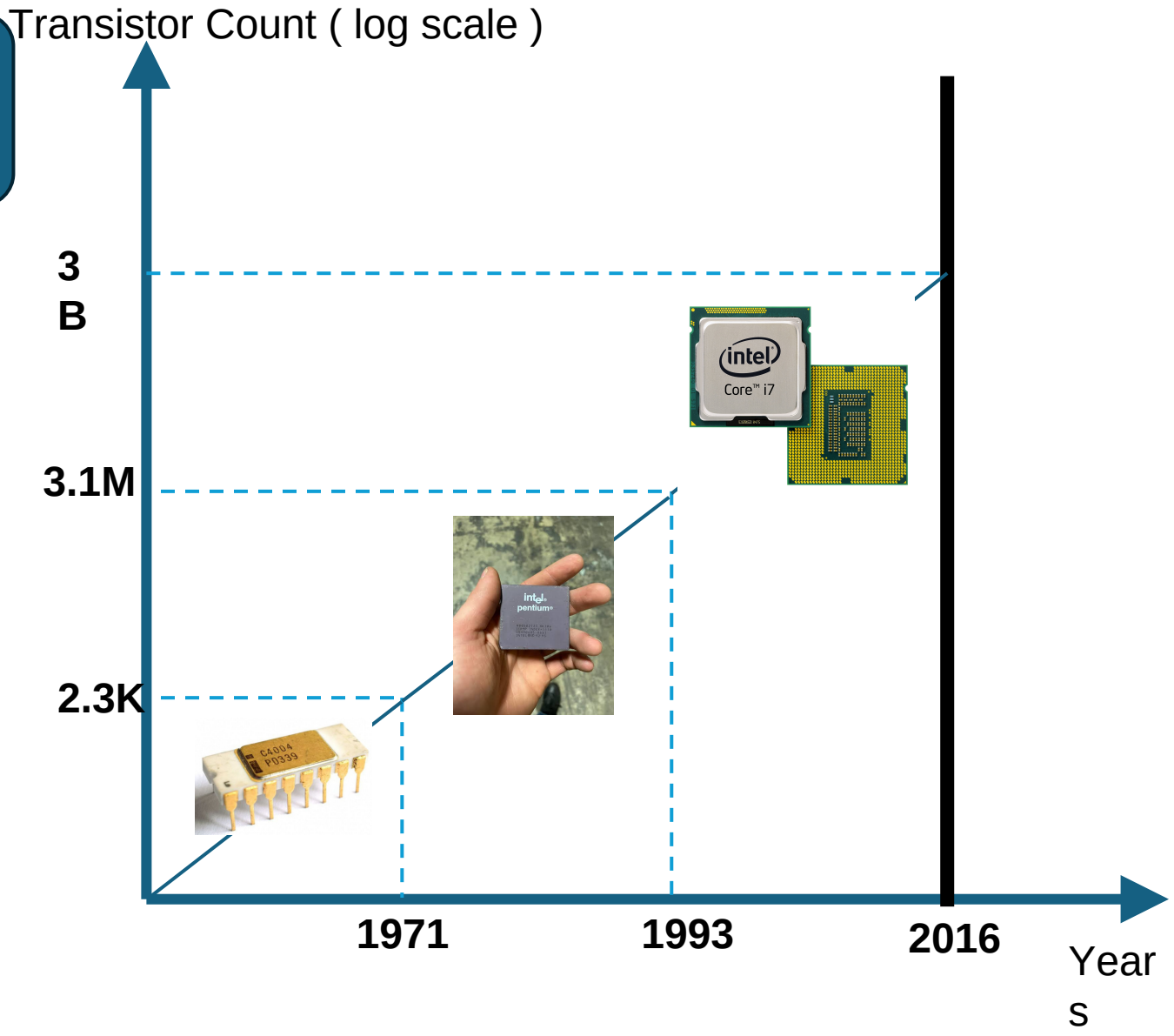
INTRODUCTION

The Problem
Limits of Moore's Law

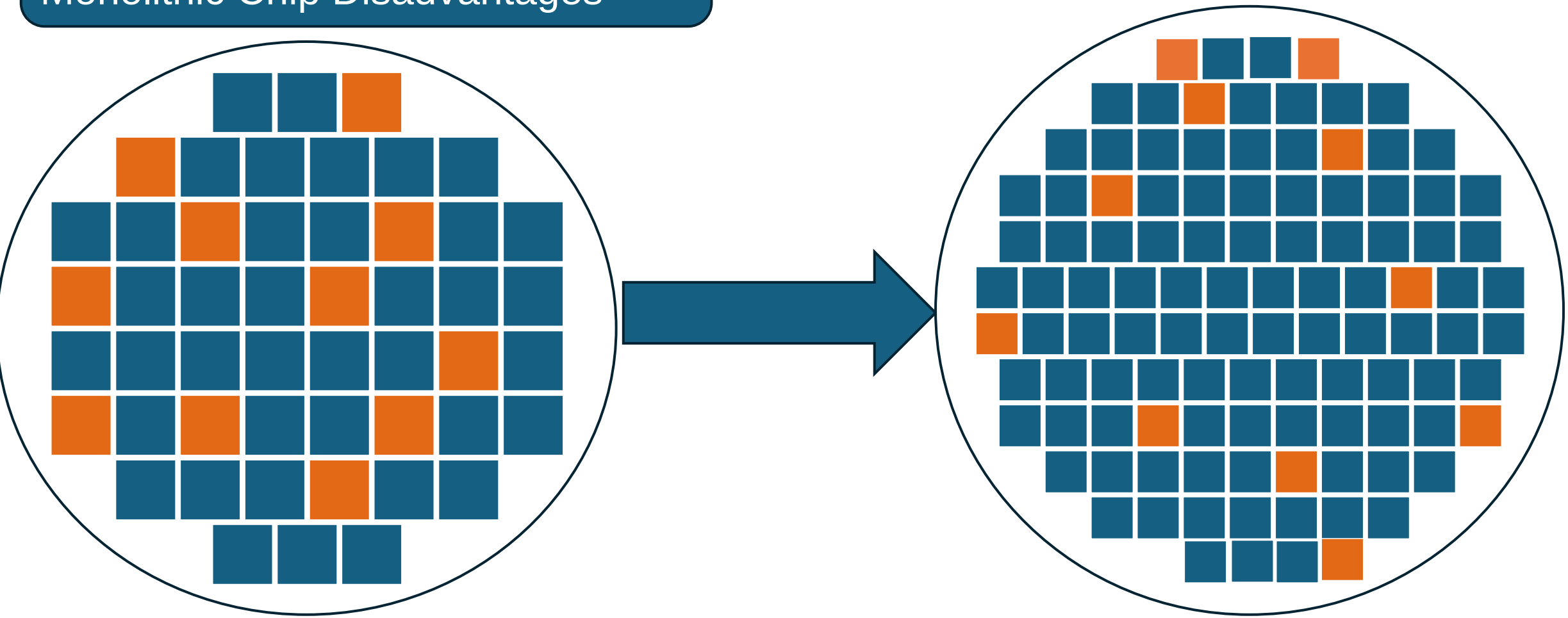
Moore's Law: Transistor count on chips doubled ~every 2 years, driving decades of continuous improvement in speed, cost, and size.

The golden era: This predictable scaling fuelled the entire semiconductor industry — smaller nodes meant faster, cheaper, and more powerful chips.

Scaling hit a wall: At 5 nm / 3 nm nodes, physical and manufacturing limits made further shrinking exponentially complex and expensive.



The Problem
Monolithic Chip Disadvantages



Yield ~ 78%

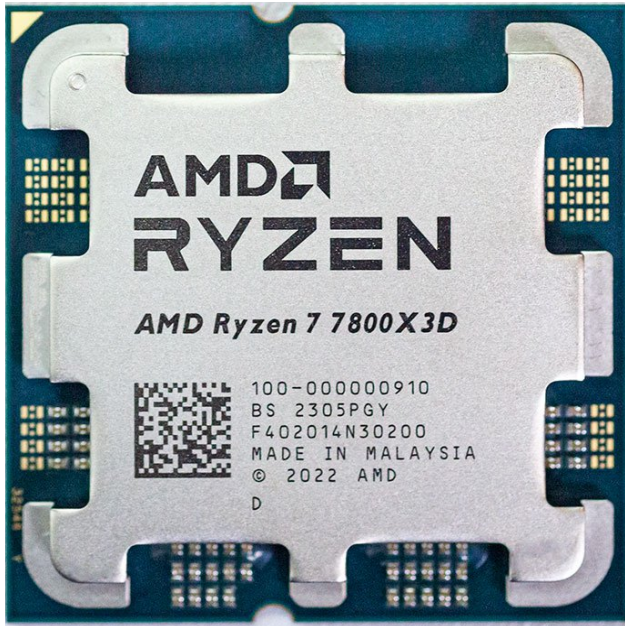
Yield ~ 90%

The Solution
Heterogeneous Integration

What is HI? Combining different chip types — logic, memory, analog, RF — into a single package.

Smart node mixing: Not every unit inside the node require the latest node

The result: Better performance, lower power consumption, and reduced overall cost.



The Solution
Heterogeneous Integration

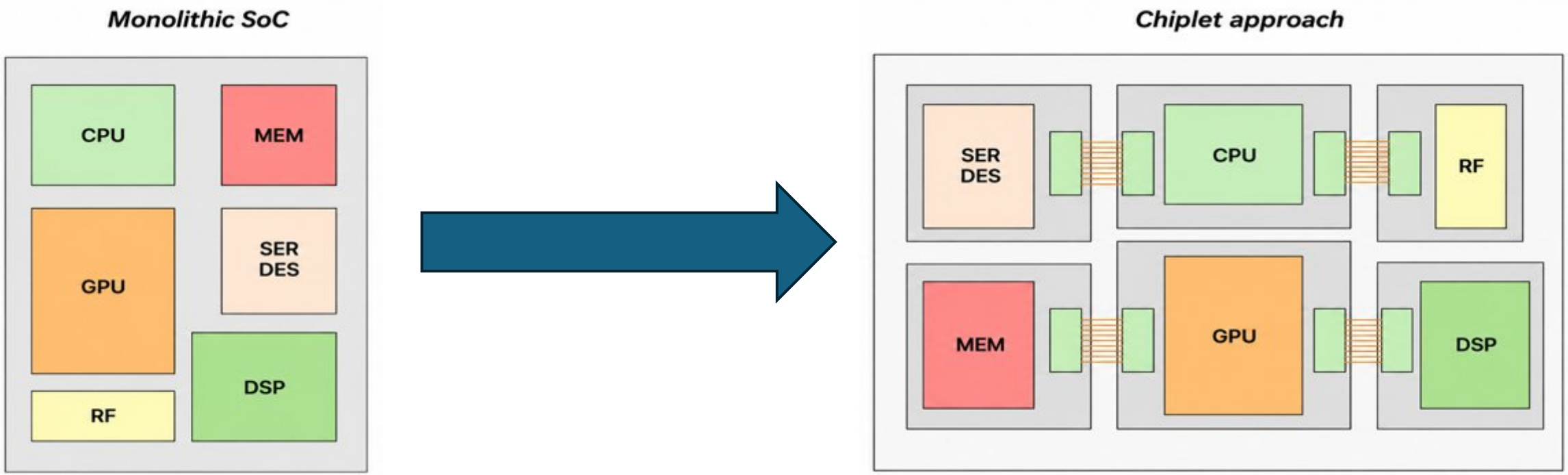
What is HI? Combining different chip types — logic, memory, analog, RF — into a single package.

Smart node mixing: Not every unit inside the node require the latest node

The result: Better performance, lower power consumption, and reduced overall cost.

Monolithic vs Chiplet

Same Functions ,Smarter Design



Chiplets Benefits

Modularity
🧩 Mix vendors & nodes

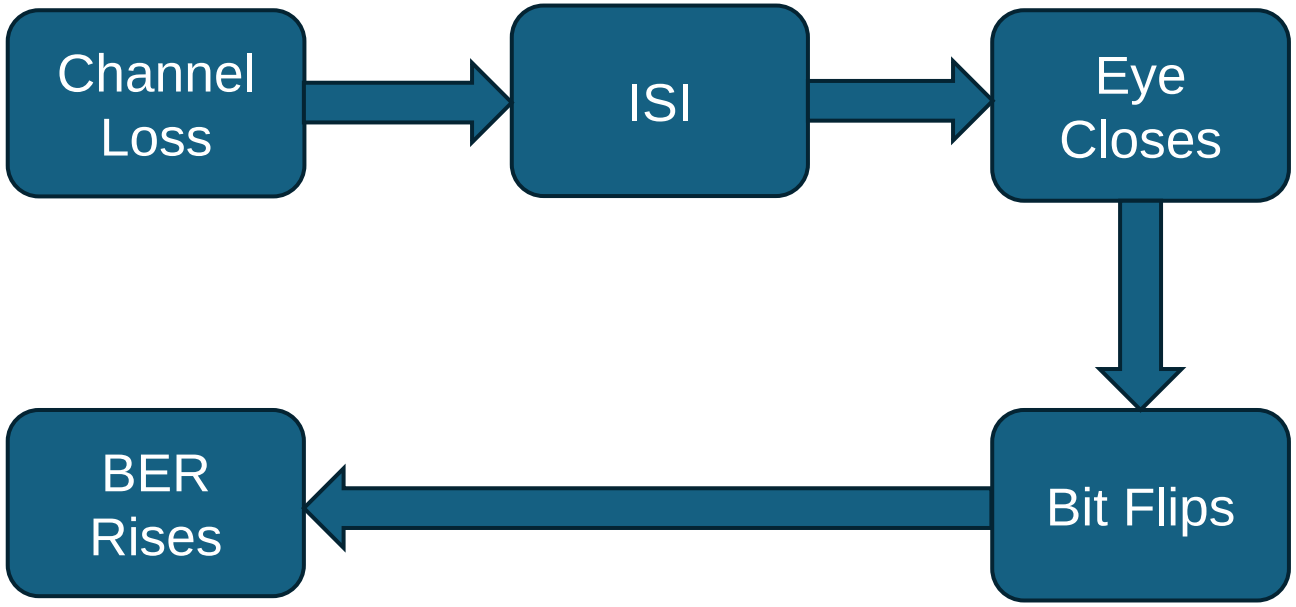
Scalability
📈 Scale without redesign

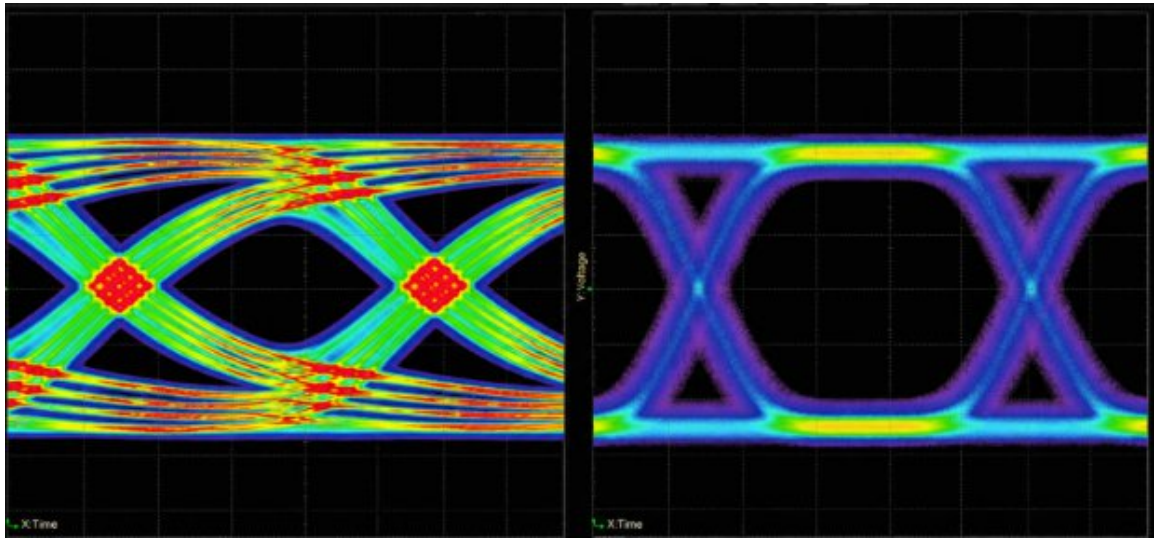
Cost Efficiency
💰 Smaller dies, higher yield

Reusability
♻ Reuse across products

Product	Company	No. of Chiplets	Impact
EPYC Processor	AMD	8	Outperformed monolithic rivals at lower cost
Ponte Vecchio	intel	47	Impossible to build as a single monolithic die
M1 Ultra		2	Doubled performance without a new process node

New Challenges
BER & ISI





With Crosstalk **Without Crosstalk**

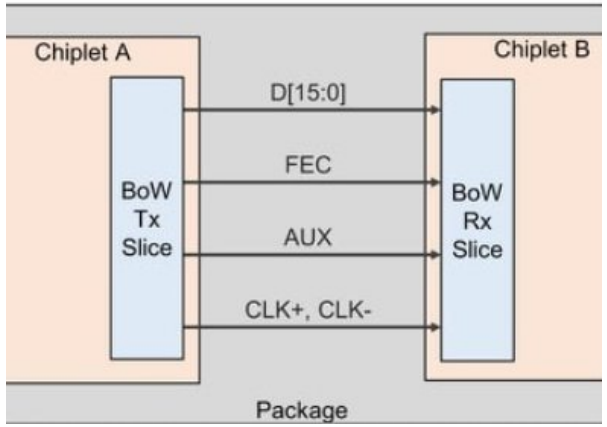
Background

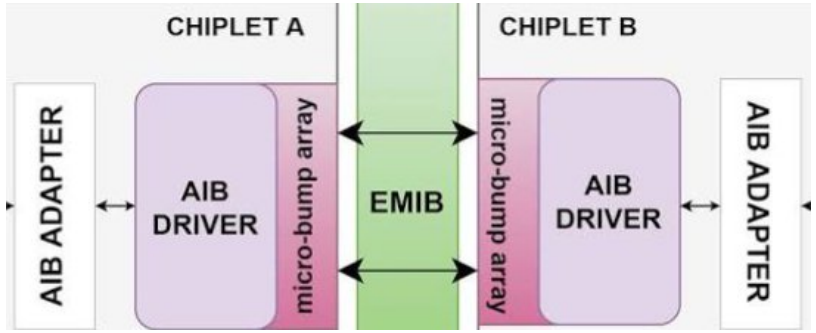
Chiplets interconnect solutions

1- BoW (Bunch of Wires)
Standard , only physical
Best for: Custom designs where the designer handles protocol adaptation and error correction.

2- AIB (Advanced Interface Bus)
Advanced , only phy
Best for: FPGA and ASIC interfacing requiring low-latency connections.

3- XSR (Extra Short Reach)
Best for: Maximum bandwidth density; requires external logic for data framing and management.

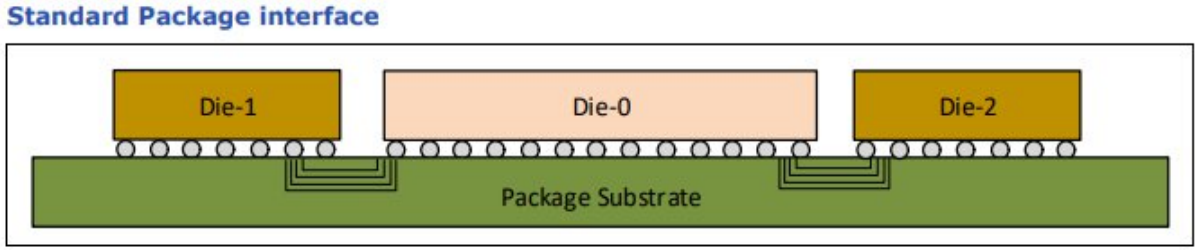




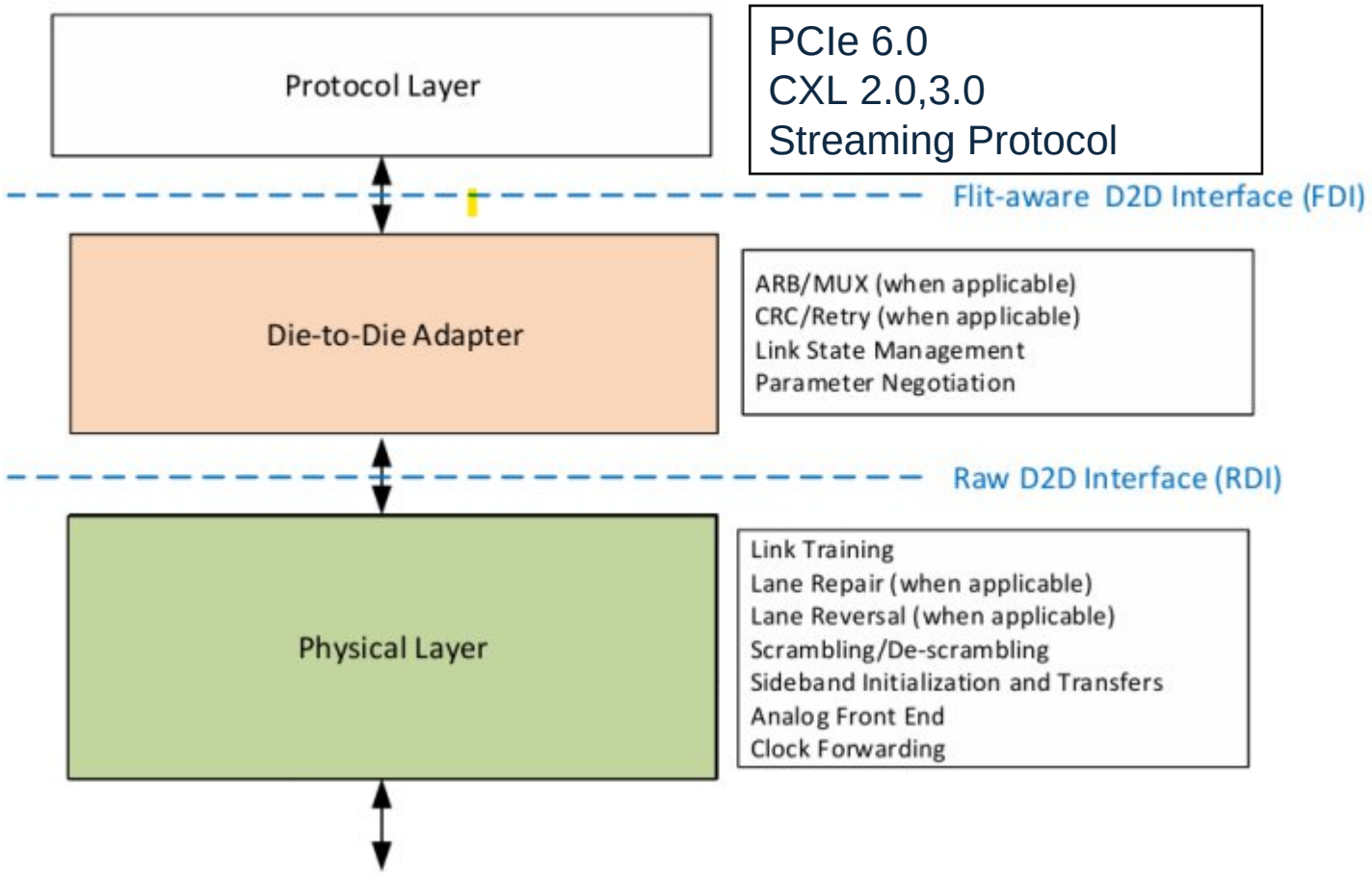
Chiplets interconnect solutions

4- UCIe (Universal Chiplet Interconnect Express)

- Most suitable solution
- Full stack communication protocol PHY + Die-to-Die Adapter + Protocol
- Standard + Advanced package, both covered Interoperable, multi-vendor, PCIe/CXL mapping
- High bandwidth density, low latency/power



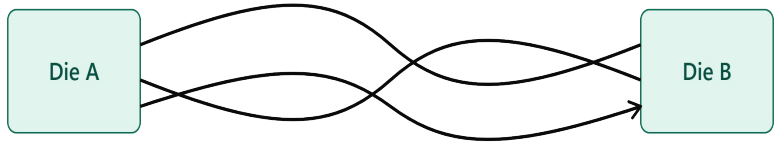
UCIe standard Architecture



PCI EXPRESS

CXL Compute Express Link

Continuous flow, no packet boundaries



Streaming protocol carries raw bit streams over UCIe with minimal overhead

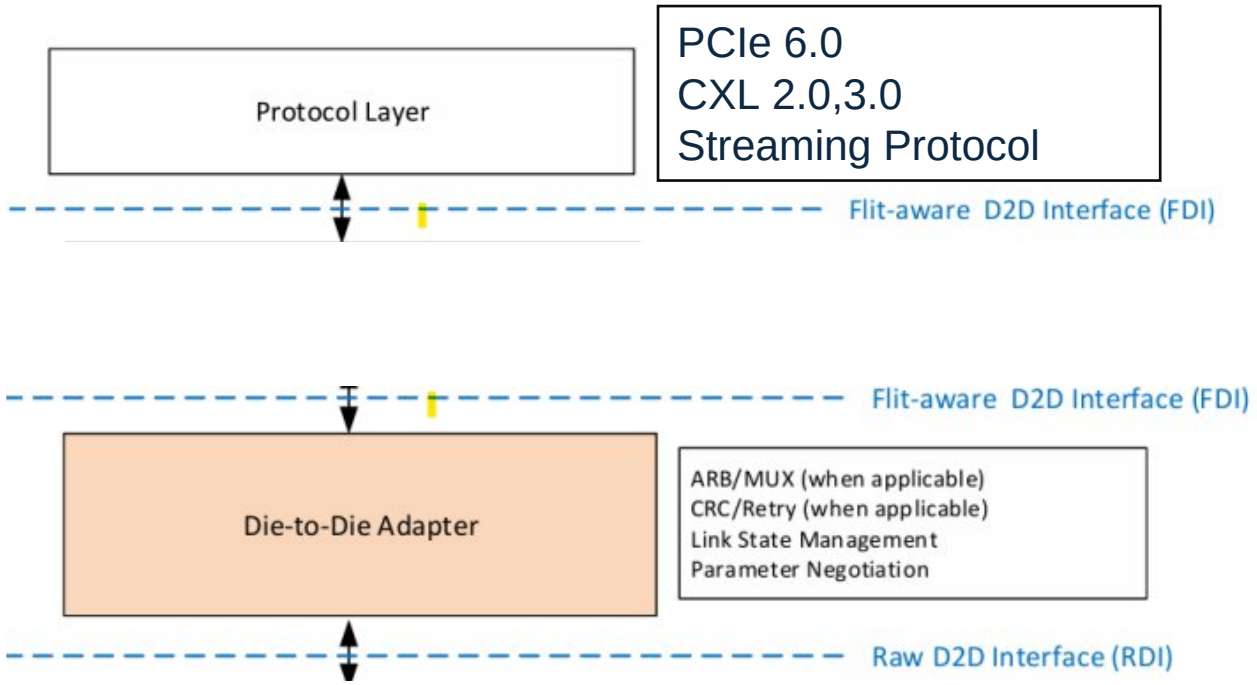
UCIe standard Architecture

1- Protocol Layer

- It is the top layer of the UCIe.
- It maps many standard protocols for enabling different applications.
- It supports PCIe 6.0, CXL 2.0, 3.0 and streaming protocols.

2- D2D Adapter layer

- Controls protocol layer and physical layer
- performs Link State Management, Parameter Negotiation and CRC/Retry .



UCIe standard Architecture

3- Physical Layer

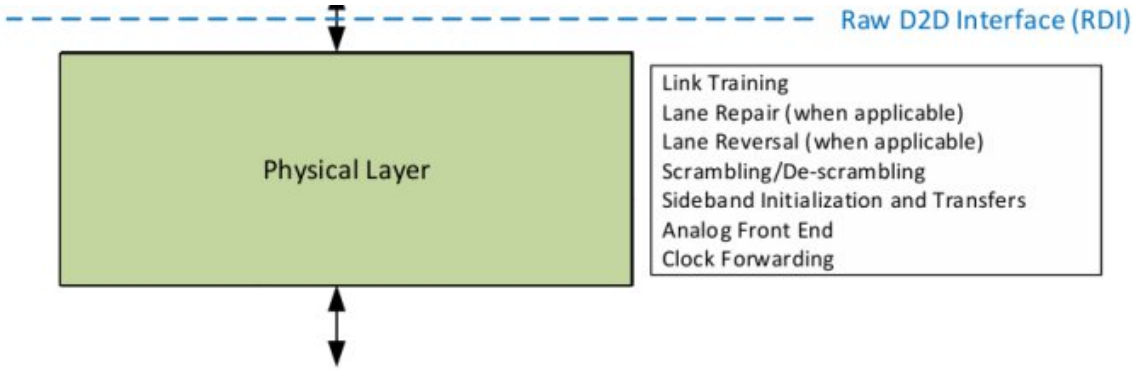
- It is the lowest layer that coordinates the different functions and their relative sequencing for proper Link bring-up and management.

- It supports the following connection types:

Side Band: Mainly used for parameter exchange, link training, and initialization.
Main Band: Main data path for protocol data transfer.

- It supports standard and advanced packaging.

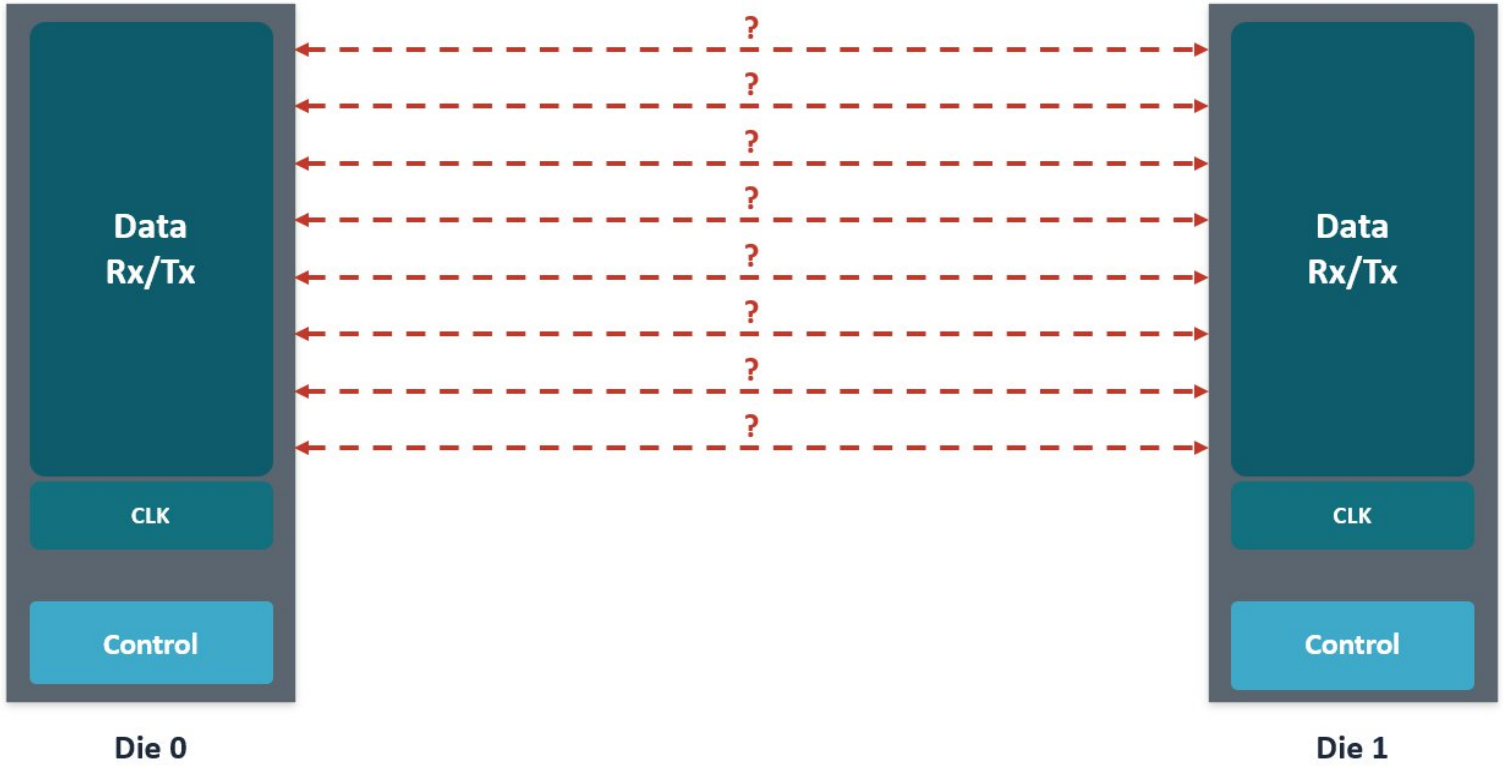
- It supports link training and lane repair.



Link Training in high speed

1- Link Initialization: RX detect

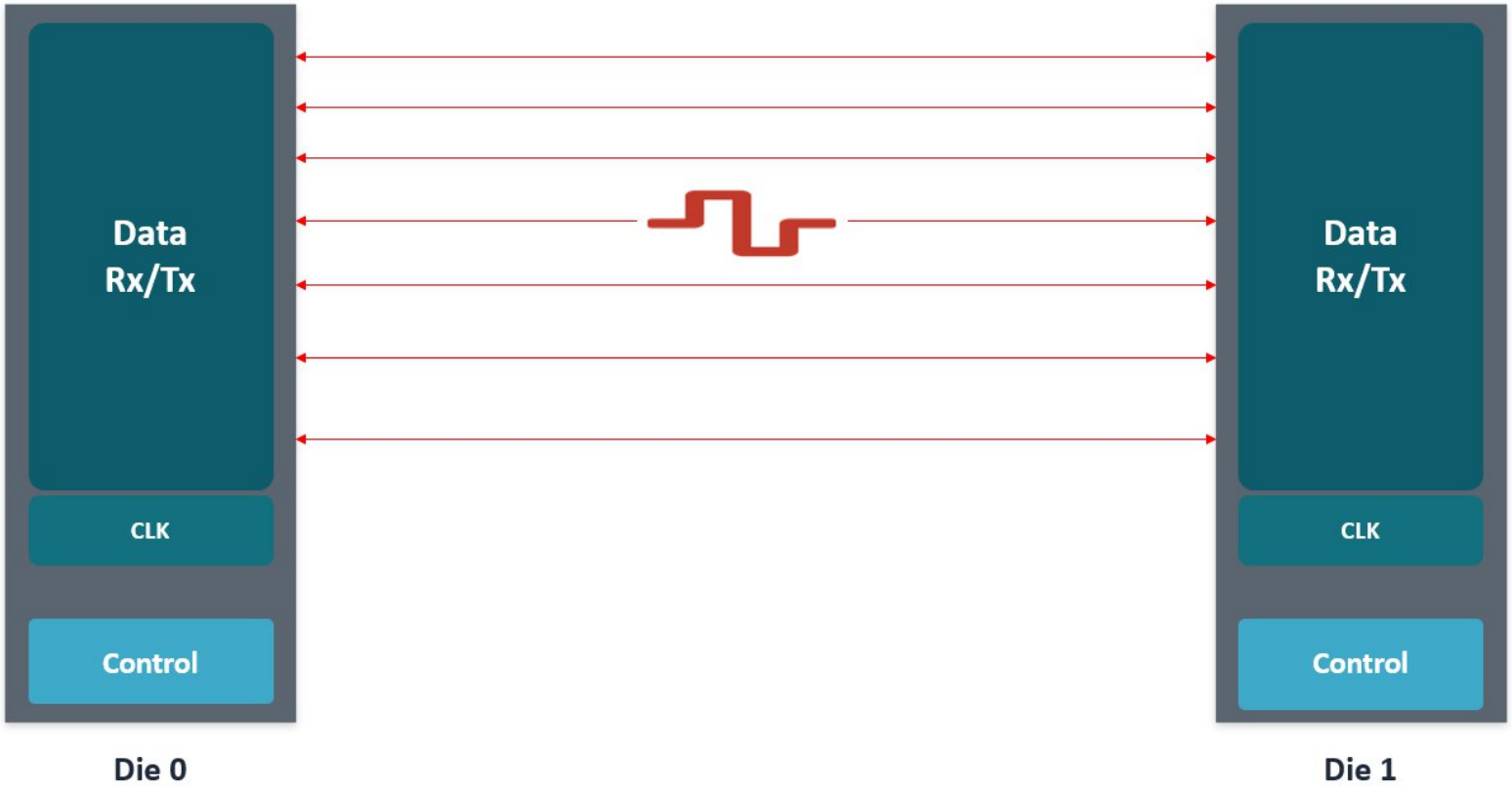
- Devices **check if a link partner is physically connected.**
- No data transmitted yet.



Link Training in high speed

2- Link Initialization: polling

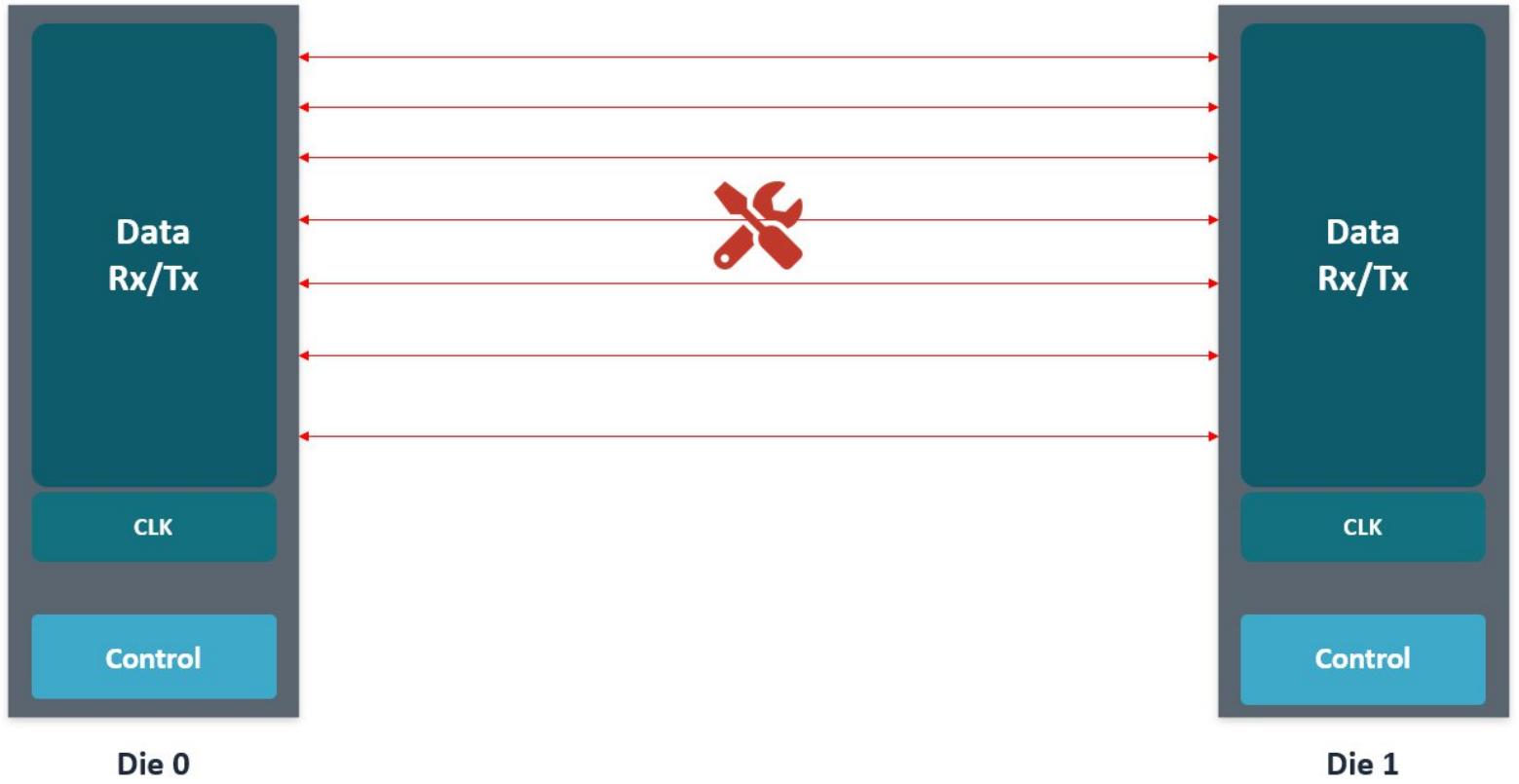
- Both sides start **sending known training patterns.**
- Used to achieve **bit/symbol lock** and **basic synchronization.**
- **Polarity and lane order get checked/fixed.**



Link Training in high speed

3- Link training: Configuration

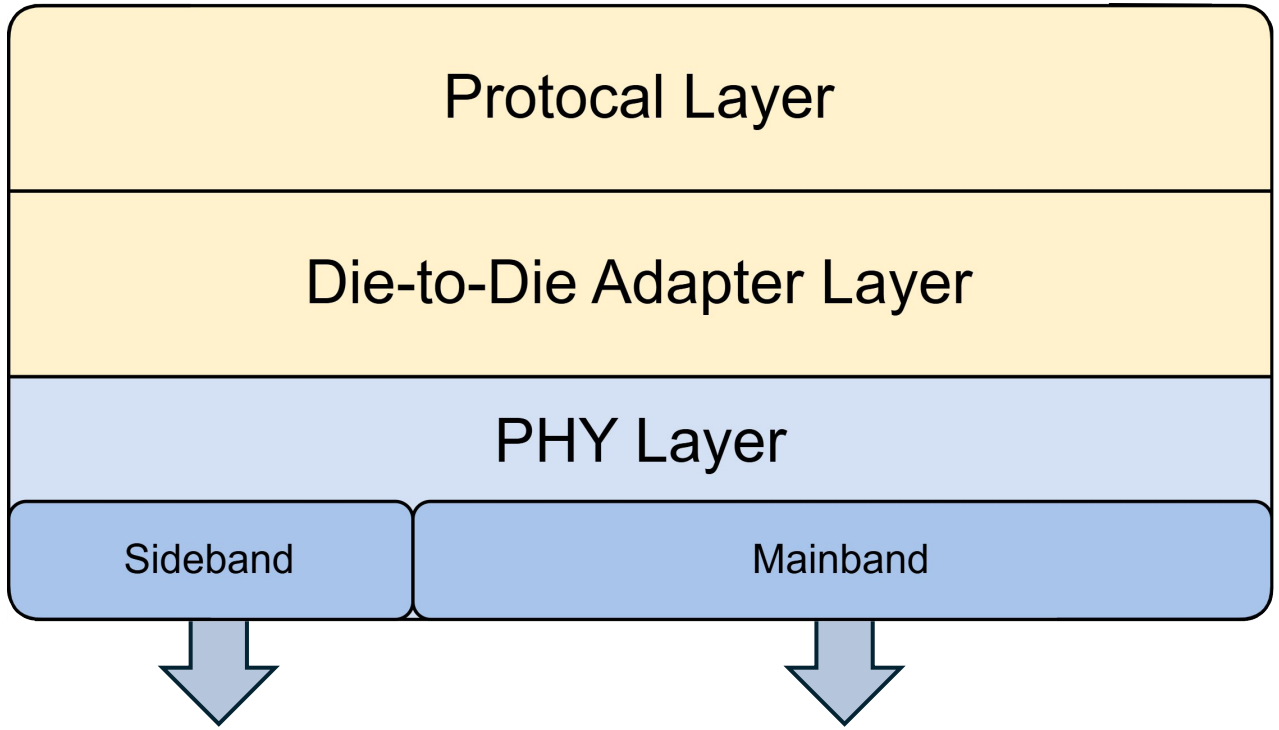
- Devices exchange and **agree on actual link parameters (width, speed, lane mapping).**
- Lanes get **grouped and deskewed** so they act as one aligned link.
- Once agreed, **link moves toward becoming active for real data**



Design Methodology

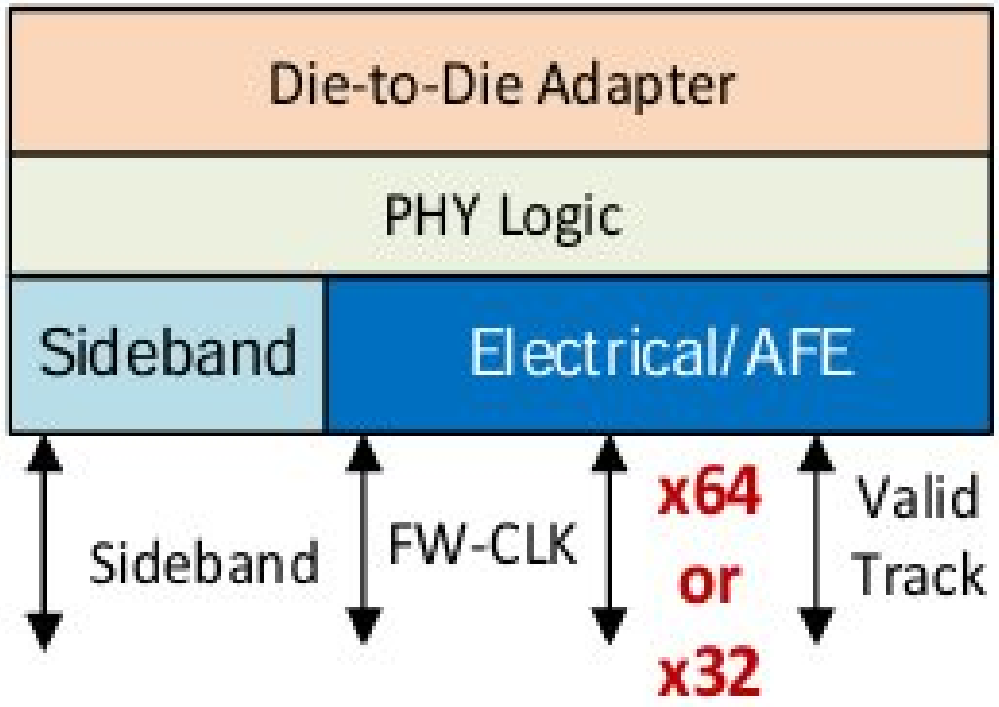
CIe PHY layer channels

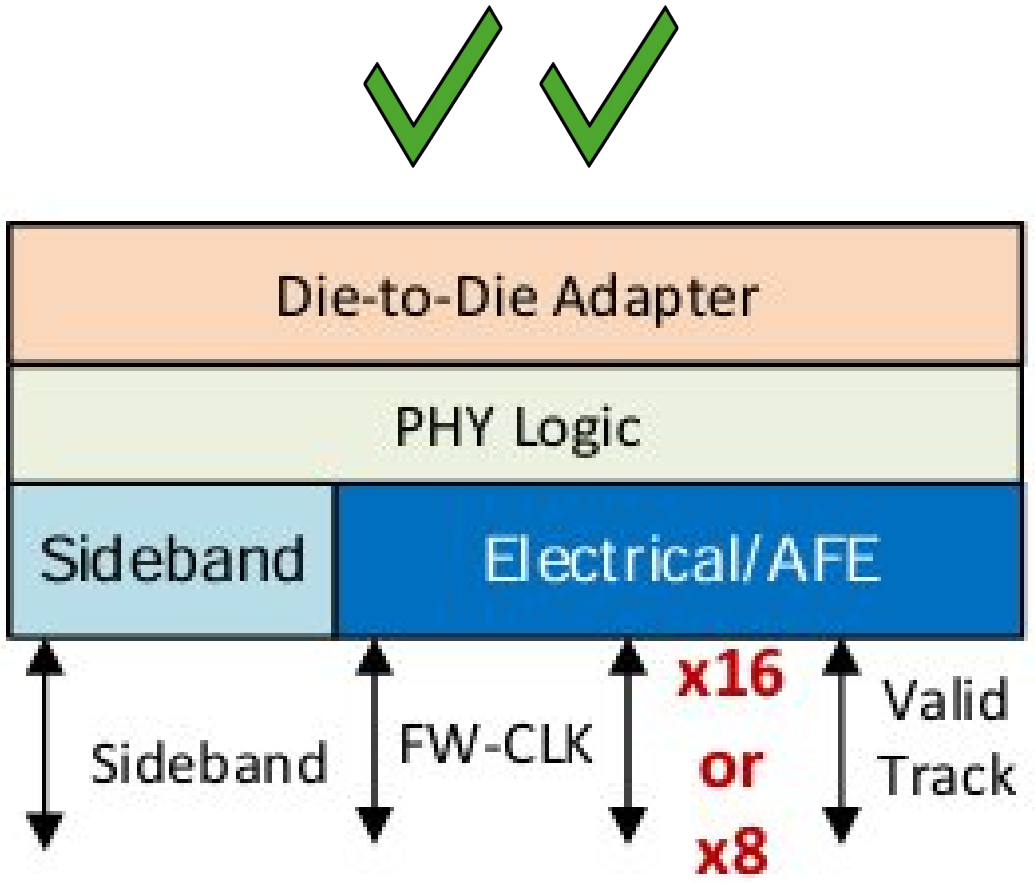
channel Vs MB channel.



CIe Configurations

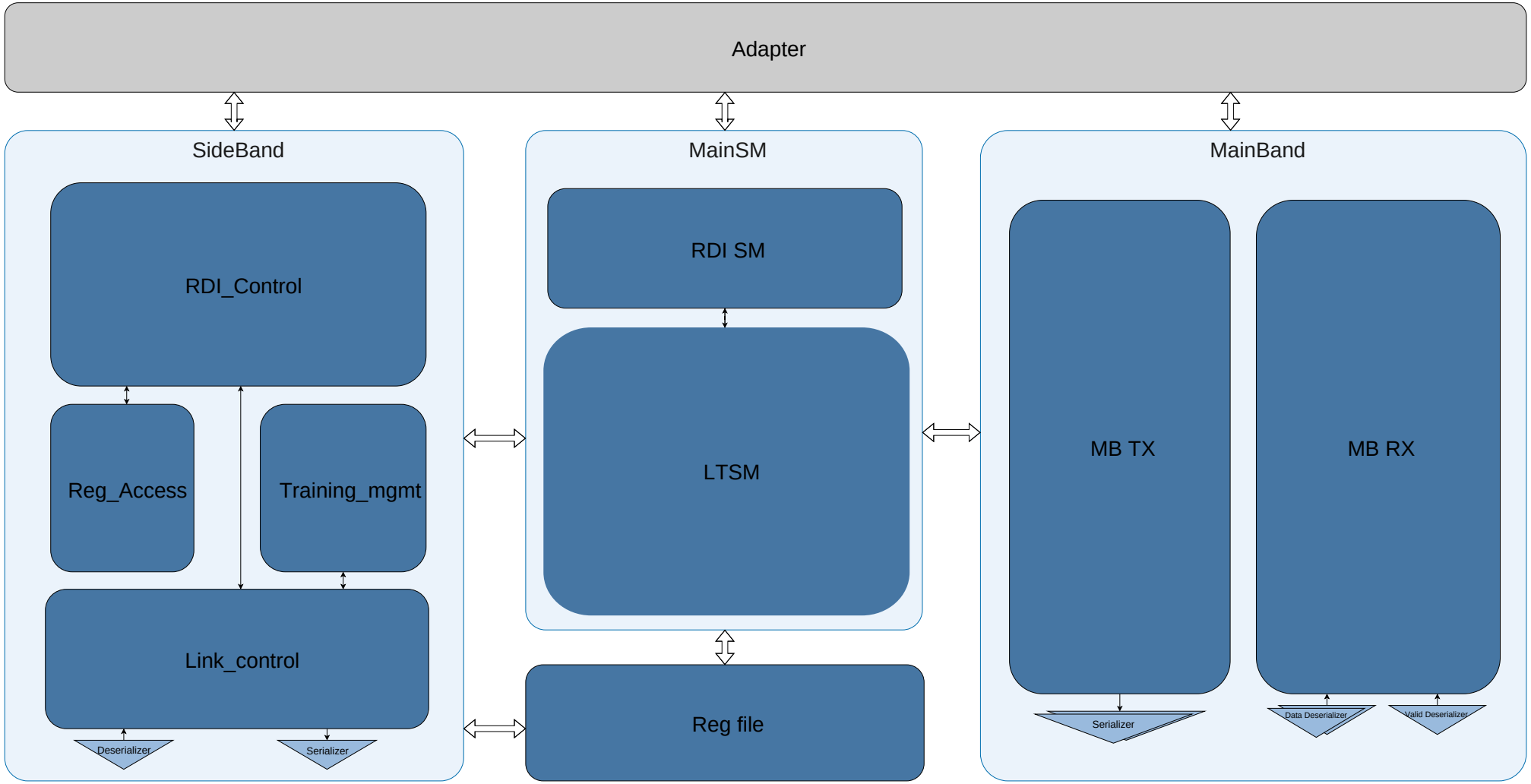
ndard Vs Advanced and Module Width



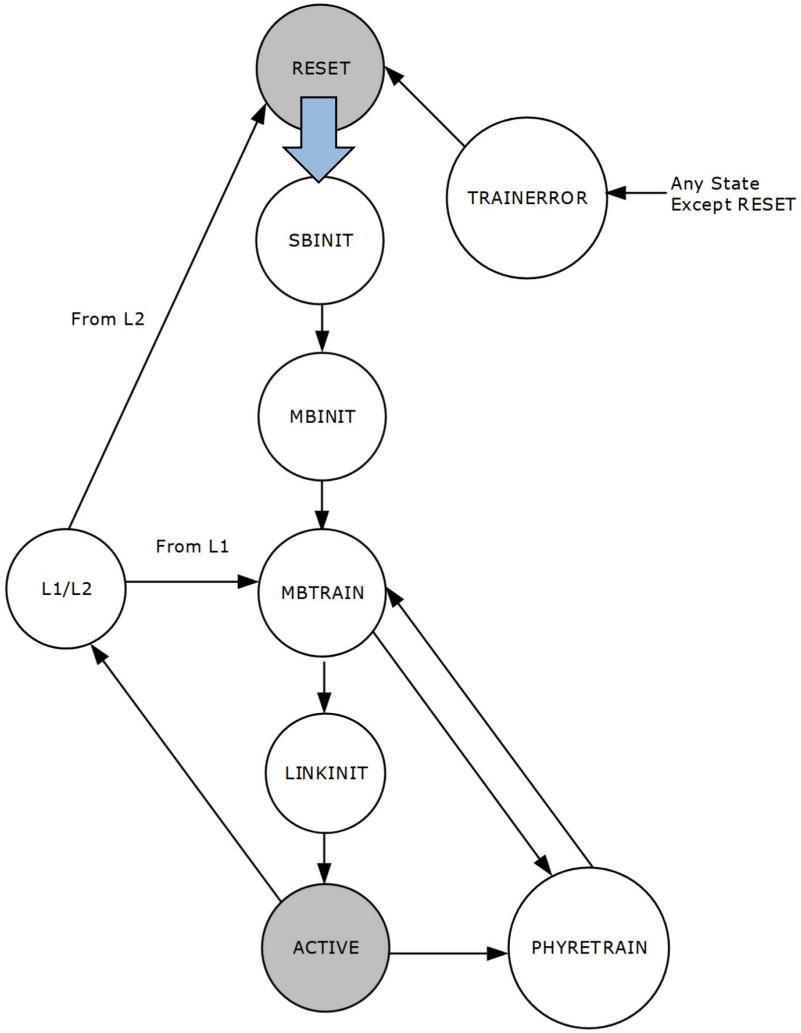


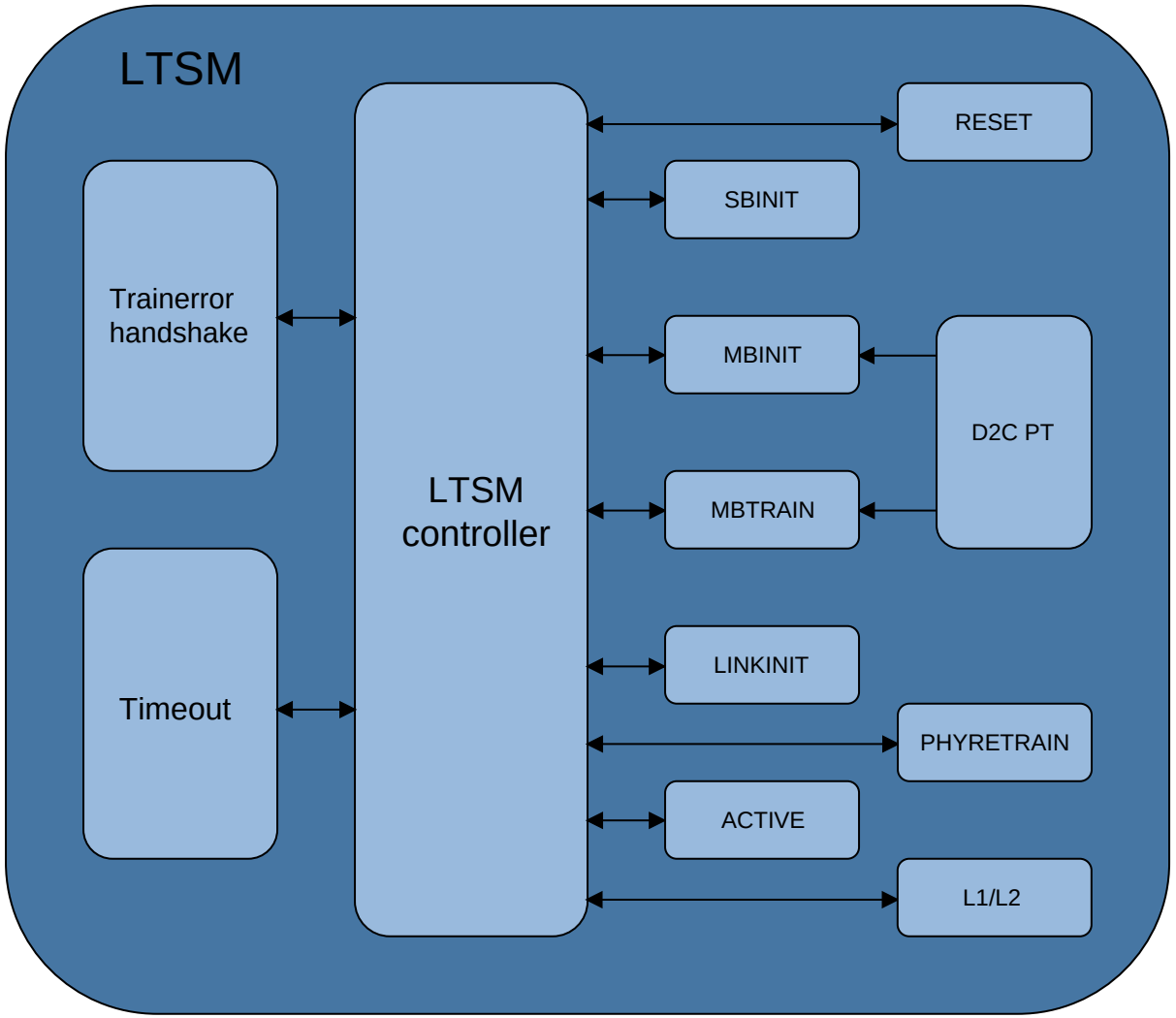
UCIe PHY Architecture

CIe PHY Architecture



Link Training State machine





RESET

Power-On / Entry State

STATE 1 / 9

RESET | SBINIT | MBINIT | MBTRAIN | LINKINIT | ACTIVE | PHYRETRAIN | TRAINERROR | L1/L2

ENTRY TRIGGERS

- Primary Reset
- or **TRAINERROR** exit

EXIT WHEN

- Power supplies stable
- Training trigger received
- Clocks ready · SB 800 MHz

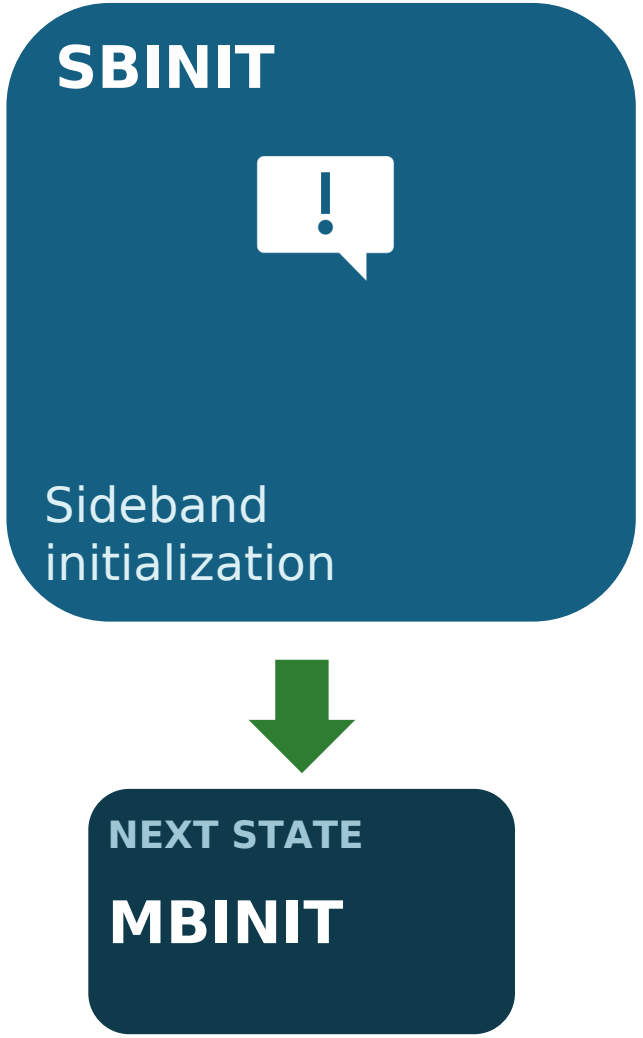
SBINIT

State Sideband detection, and Out-of-Reset transmission

STATE 2 / 9

EXIT WHEN

- Test sideband
- Done Handshake



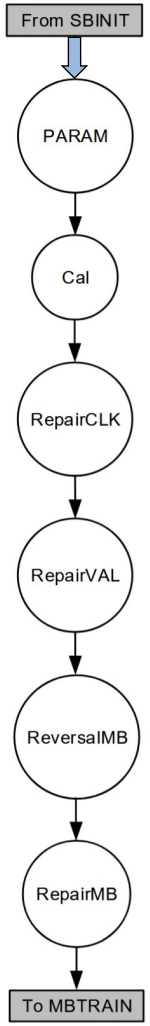
MBINIT

Mainband initialization, on-die calibration, and interconnect repair (if applicable)

STATE 3 / 9

EXIT WHEN

- Parameter exchanged
- Test MB lanes
- MBINIT.Repair.MB Done Handshake



MBINIT

Paramter Exchange and MB, clock , valid Lanes initialization

NEXT STATE

MBTRAIN

MBTRAIN

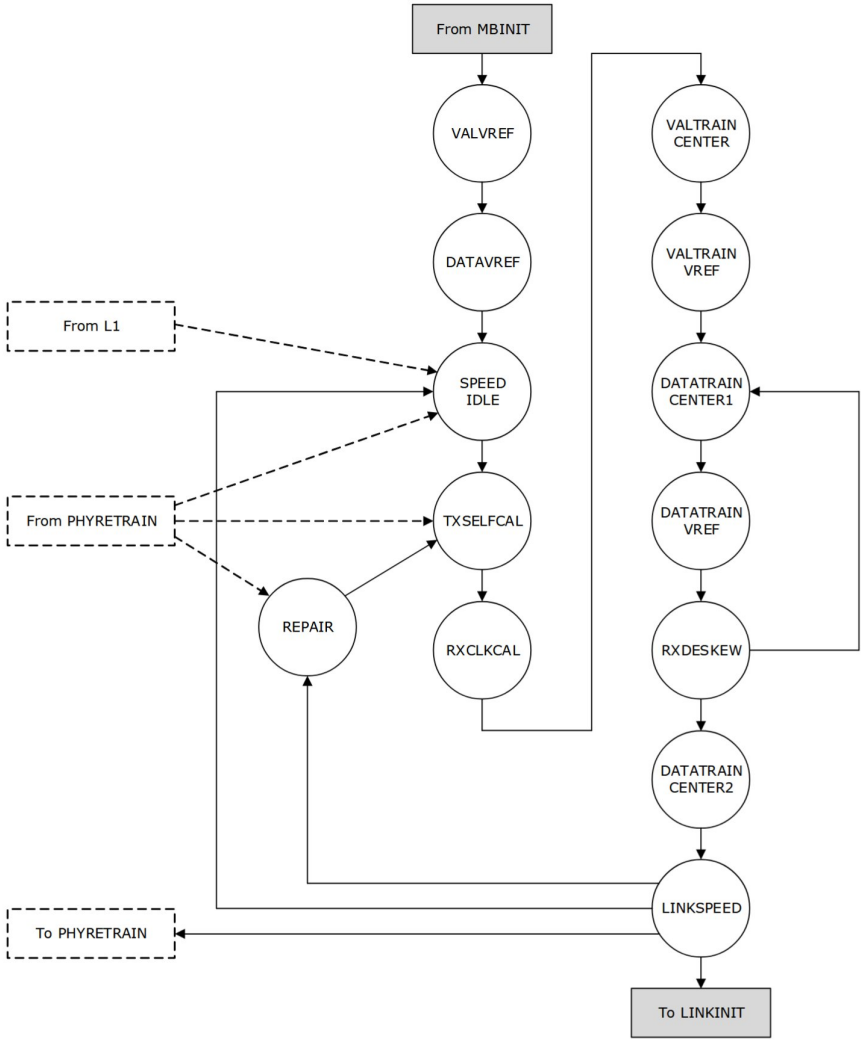
Training the MainBand at the Targeted Parameters

STATE 4 / 9

RESET | SBINIT | MBINIT | **MBTRAIN** | LINKINIT | ACTIVE | PHYRETRAIN | TRAINERROR | L1/L2

EXIT WHEN

- Exit to LINKINIT after all MBTRAIN sub-states complete successfully
- MBTRAIN.LINKSPEED Done Handshake



MBTRAIN

Configures the Data, Clock, and Valid signals to the highest negotiated speed and performs mainband training

NEXT STATE

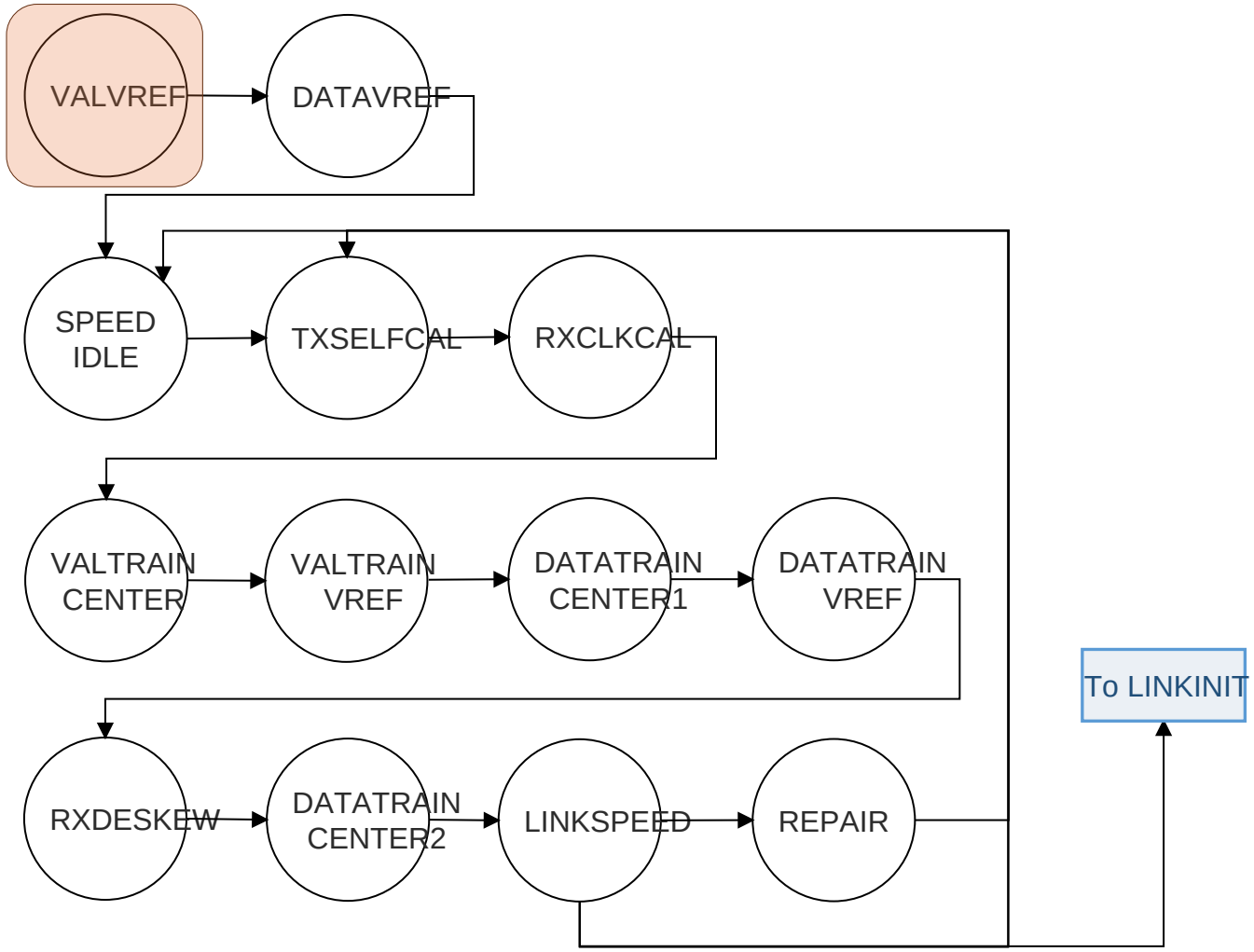
LINKINIT

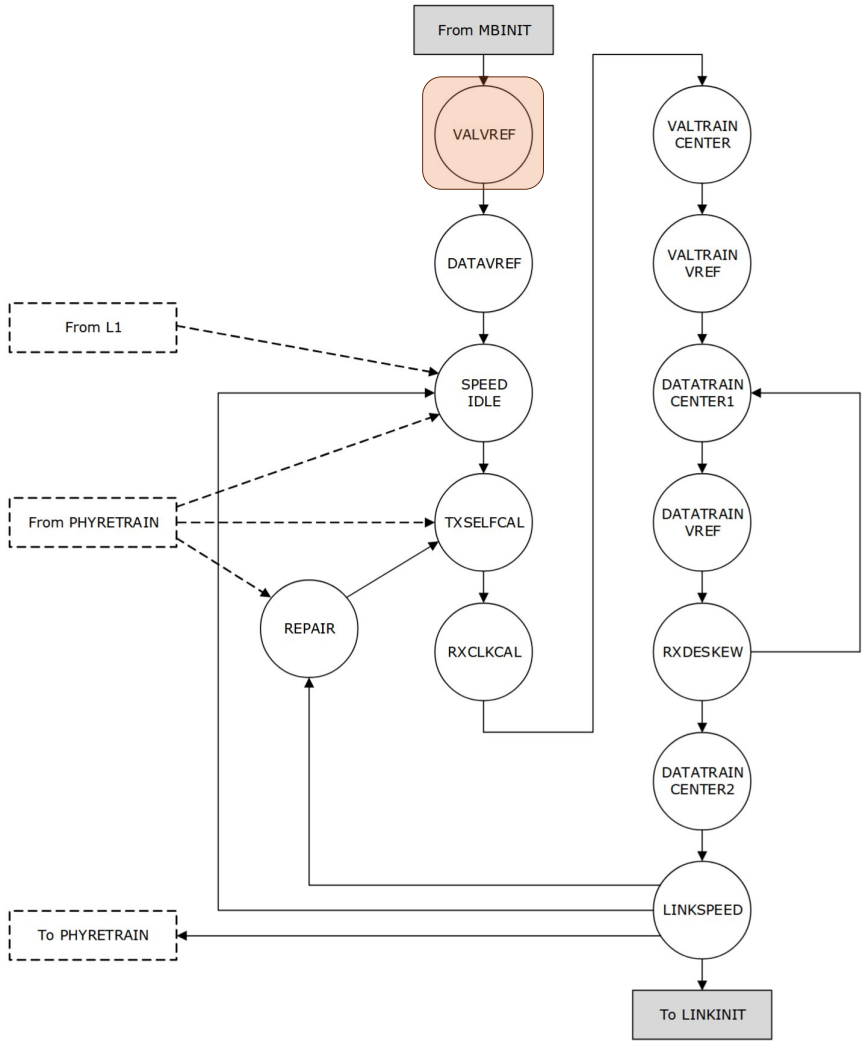
MBTRAIN

Training the MainBand at the Targeted Parameters

STATE 4 / 9

RESET | SBINIT | MBINIT | **MBTRAIN** | LINKINIT | ACTIVE | PHYRETRAIN | TRAINERROR | L1/L2





MBTRAIN

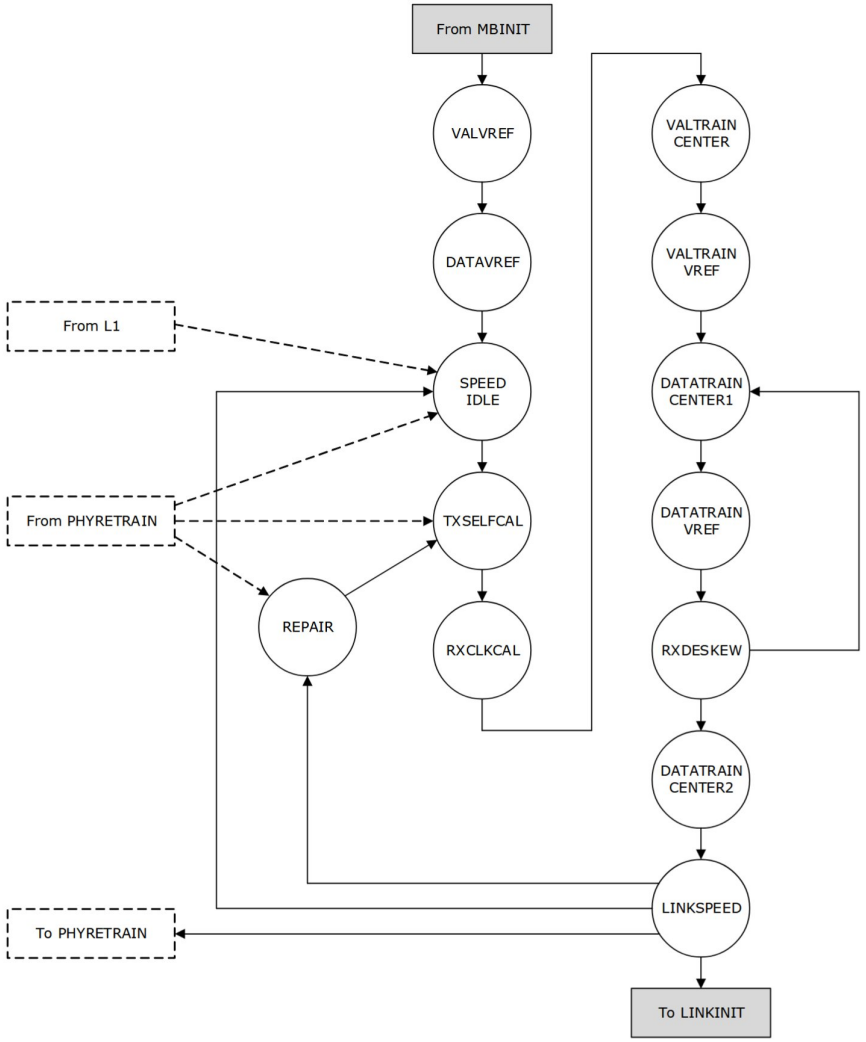
Training the MainBand at the Targeted Parameters

STATE 4 / 9

RESET | SBINIT | MBINIT | **MBTRAIN** | LINKINIT | ACTIVE | PHYRETRAIN | TRAINERROR | L1/L2

EXIT WHEN

- Exit to LINKINIT after all MBTRAIN sub-states complete successfully
- MBTRAIN.LINKSPEED Done Handshake



MBTRAIN

Configures the Data, Clock, and Valid signals to the highest negotiated speed and performs mainband training

NEXT STATE

LINKINIT

LINKINIT

Waiting for Adapter acknowledgement to go to ACTIVE state

STATE 5 / 9

EXIT WHEN

- RDI reaches the Active state after successful adapter link initialization and successful Active handshake.

LINKINIT

Permit adapter to exchange management message

NEXT STATE

ACTIVE

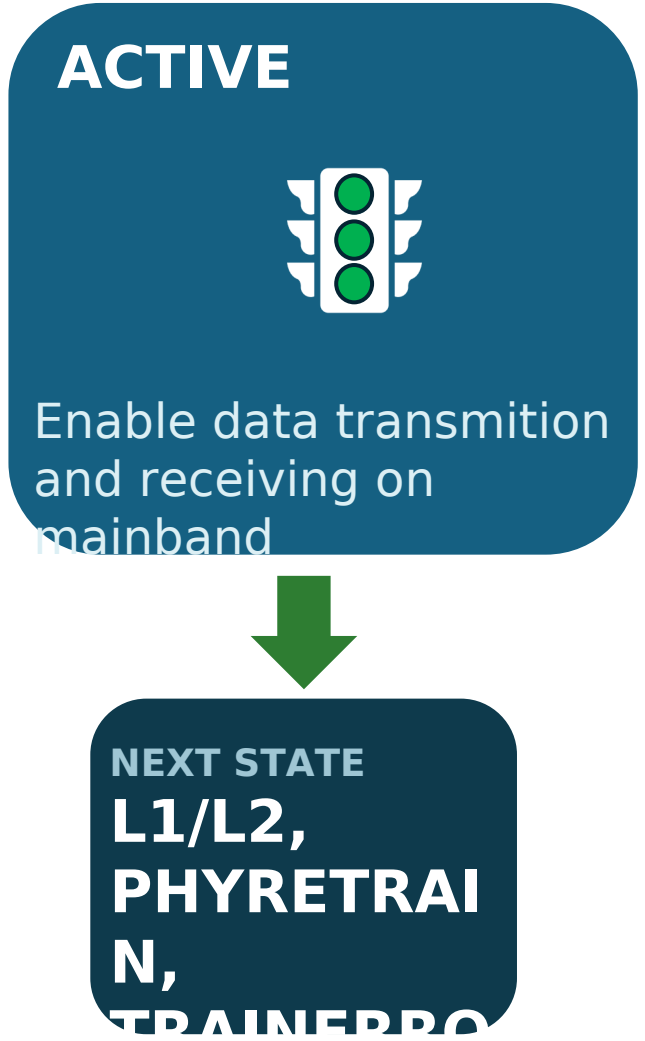
ACTIVE

This is the state in which transactions are sent and received.

STATE 6 / 9

EXIT WHEN

- Adapter Requests **Reset** on RDI
- Adapter Requests **Retrain** on RDI



PHYRETRAIN

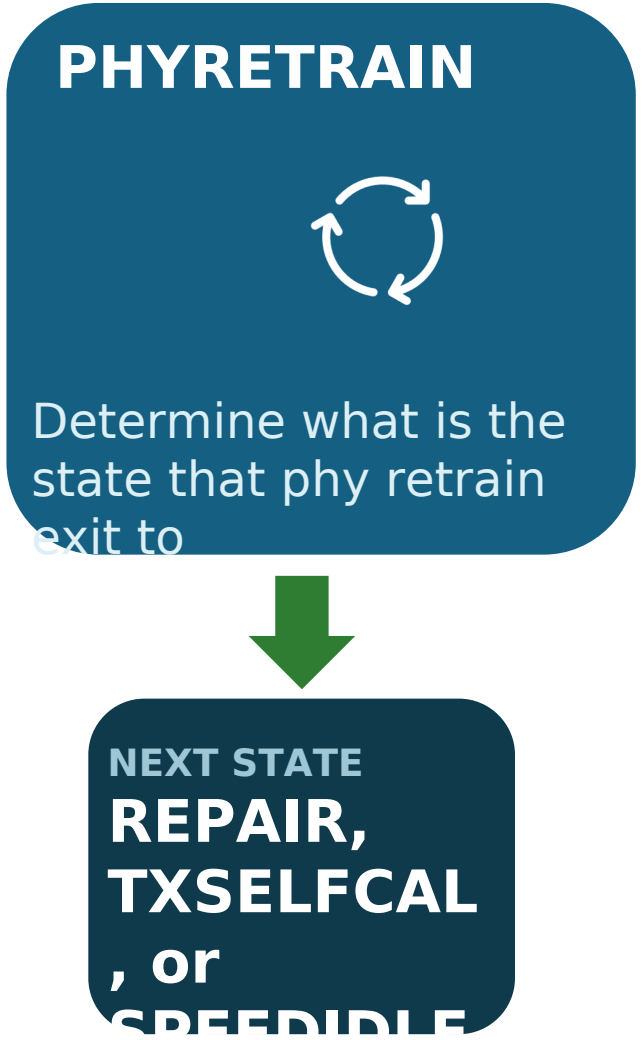
This state is used to begin the retrain flow for the Link during runtime.

STATE 7 / 9

RESET | SBINIT | MBINIT | MBTRAIN | LINKINIT | ACTIVE | **PHYRETRAIN** | TRAINERROR | L1/L2

EXIT WHEN

- When PHYRETRAIN start Handshake is done



TRAINERROR

This state entered when a fatal or non-fatal event that need to bring the state machine back to RESET state, such as a timeout.

STATE 8 / 9

ENTRY TRIGGERS

- **Trainerror andshake**
  (local or remote)

EXIT WHEN

- Adapter permit ltsm to go to reset
- or after waiting 8ms

TRAINERROR

We enter this state when train error hand shake done until Adapter permit LTSM to go to reset
Tristate all mainband lanes

NEXT STATE

RESET

L1/L2

This state entered when RDI has transitioned to PM state for power saving features.

STATE 9 / 9

ENTRY TRIGGERS

- RDI power managment

EXIT WHEN

- When the **local Adapter** requests Active on RDI or **remote Link partner** requests L1 exit (exit to **MBTRAIN.SPEEDIDLE**)
- When the **local Adapter** requests Active on RDI or **remote Link partner** requests L2 exit (exit to **RESET**)

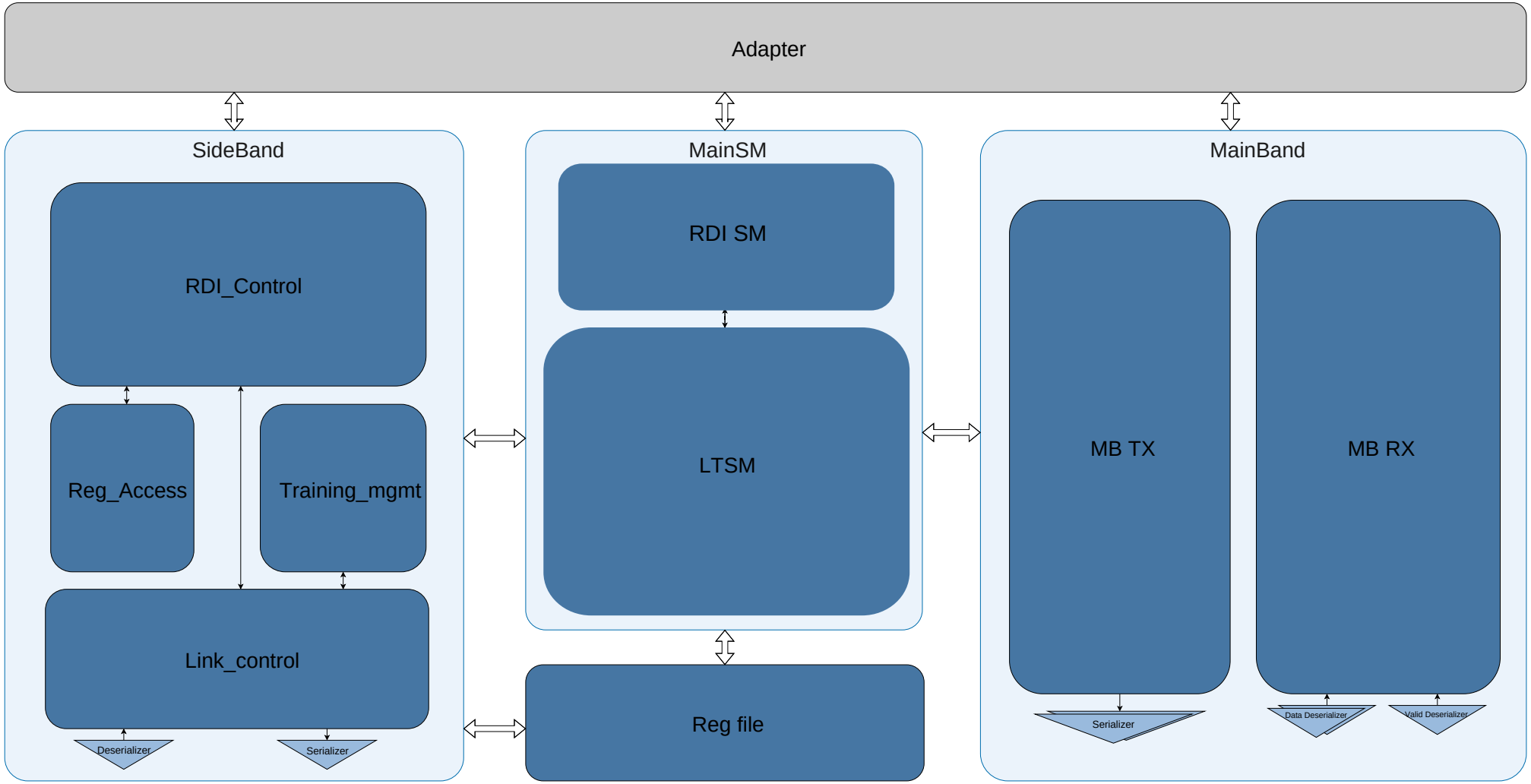
L1/L2

Tri-state Data, Valid, Clock, and Track lanes in the Mainband.

NEXT STATE

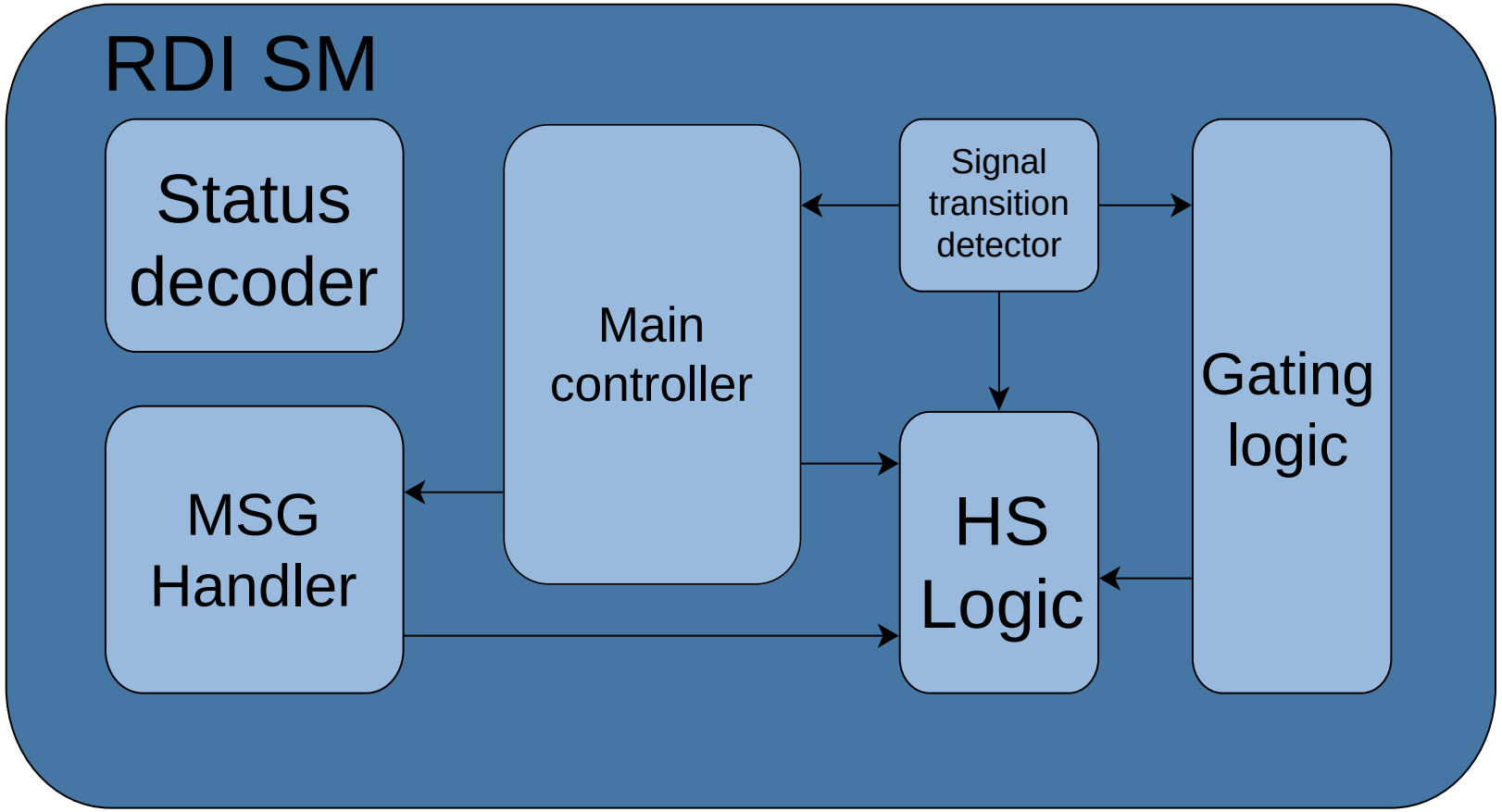
SPEEDIDLE, RESET

Cle PHY Architecture



RDI SM

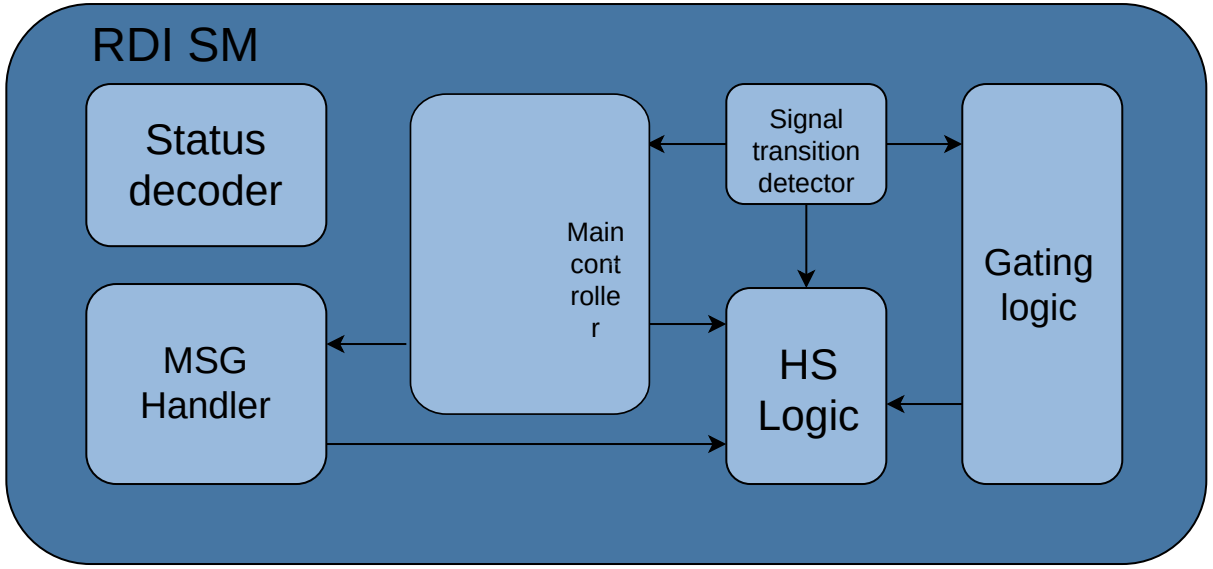
Link Management and Handshakes Logic



RDI SM
Link Management and Handshakes Logic

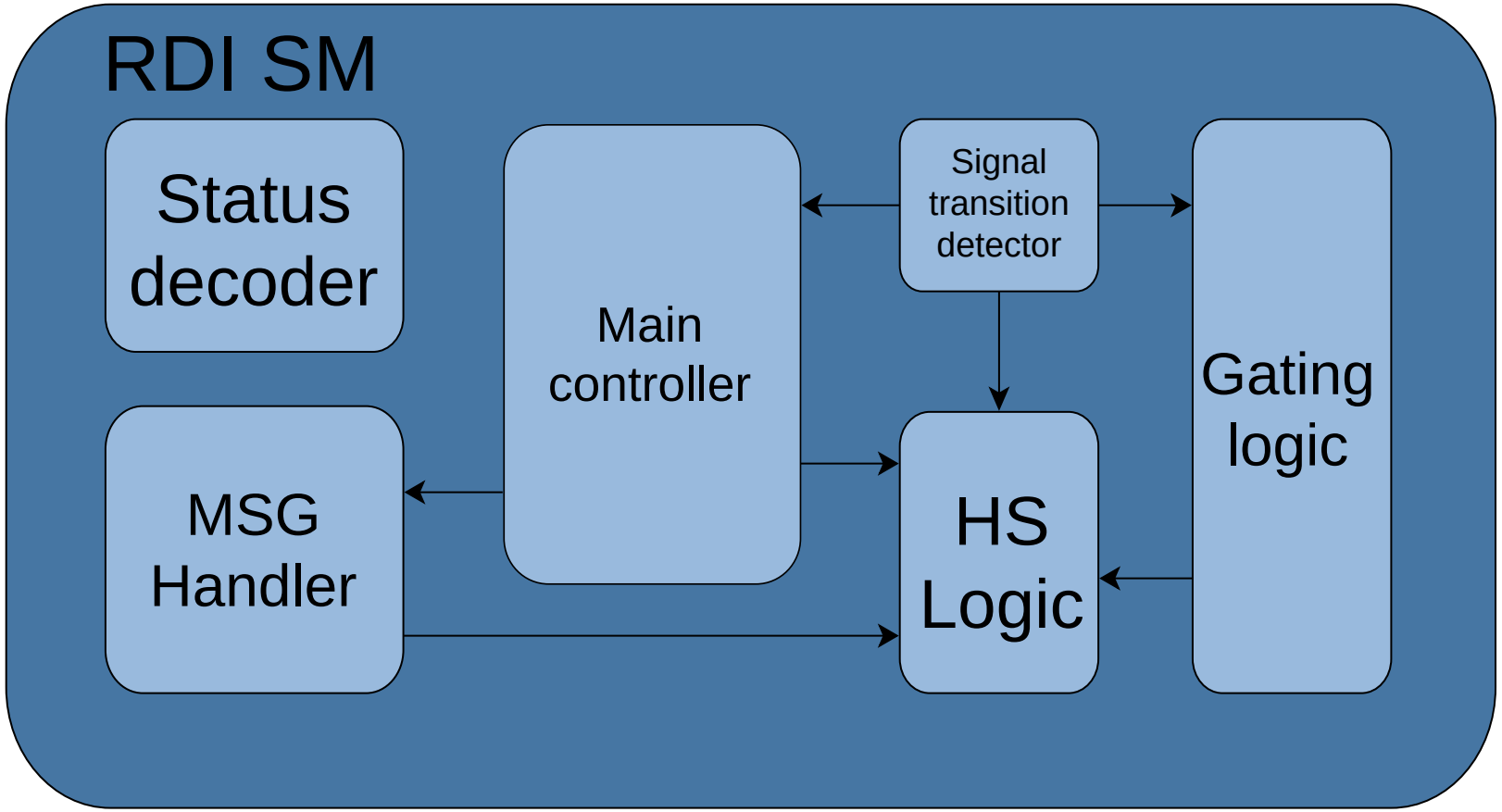
RDI SM — Main Functions

- Abstracts PHY state to adapter.
- runs RDI FSM .
- Coordinates state with remote partner via `{LinkMgmt.RDI*}` sideband messages.
- Adapter handshakes: clk, wake, stall, Active.
- Power management.
- Retrain handling; hands training to LTSM.



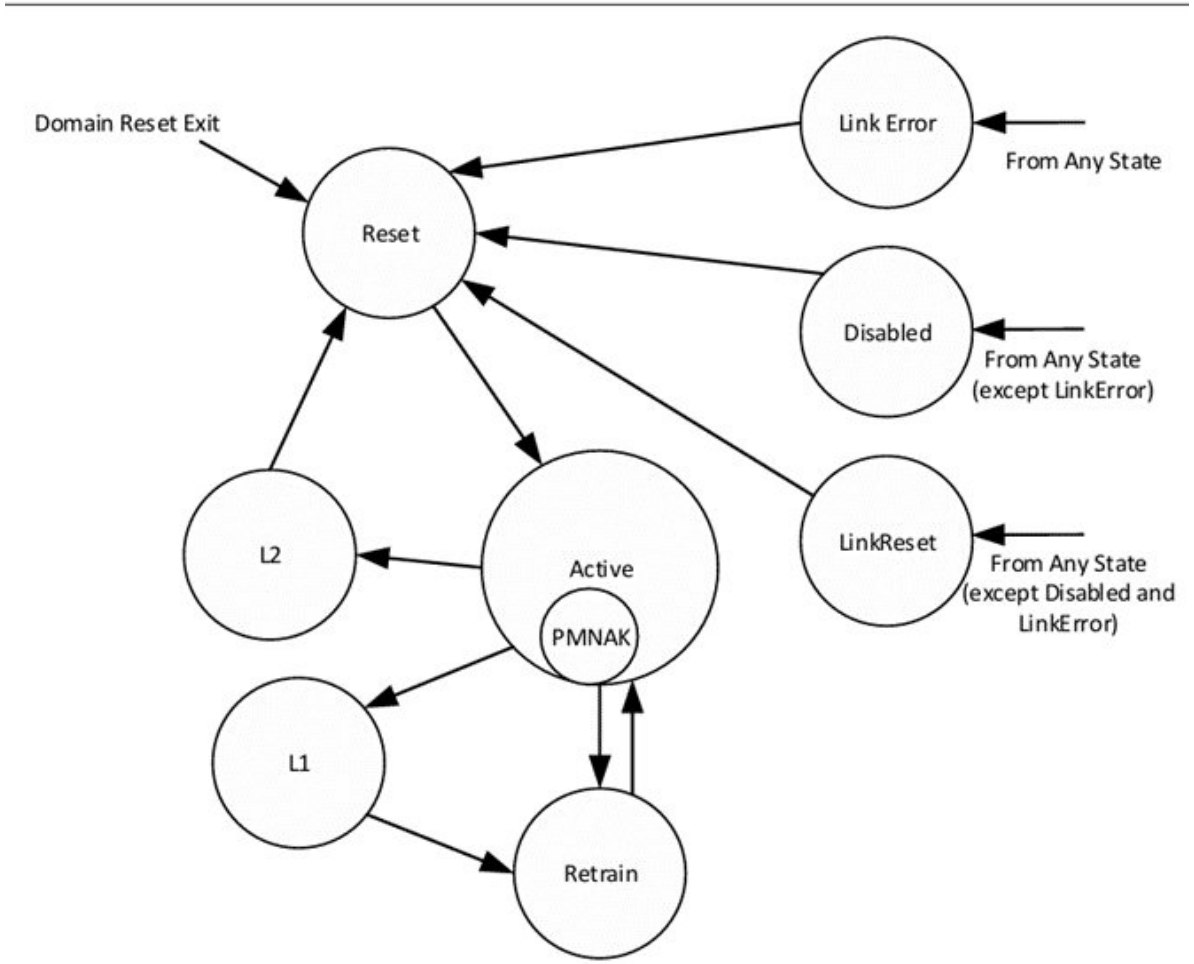
RDI SM

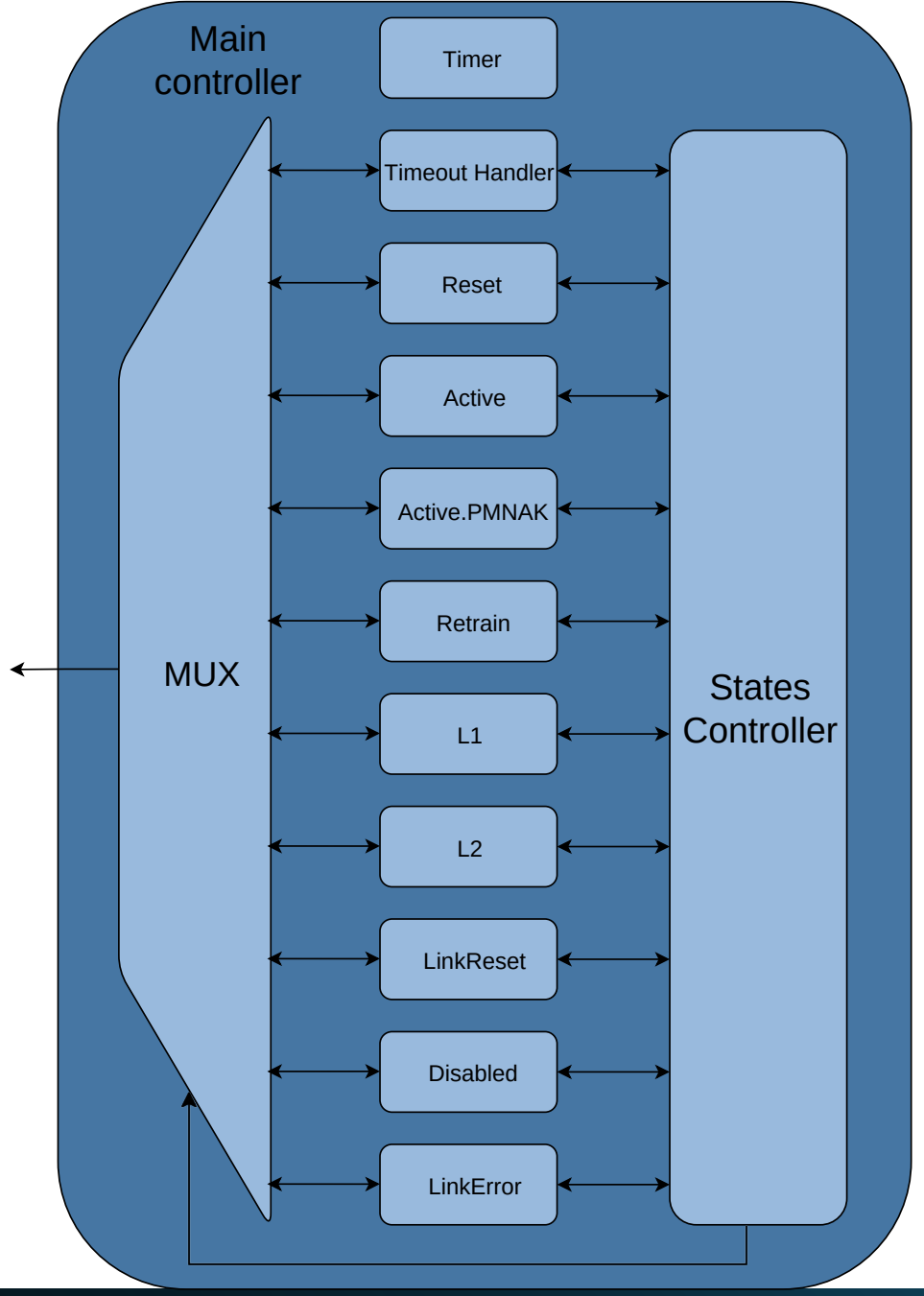
Link Management and Handshakes Logic



RDI SM

Main Controller



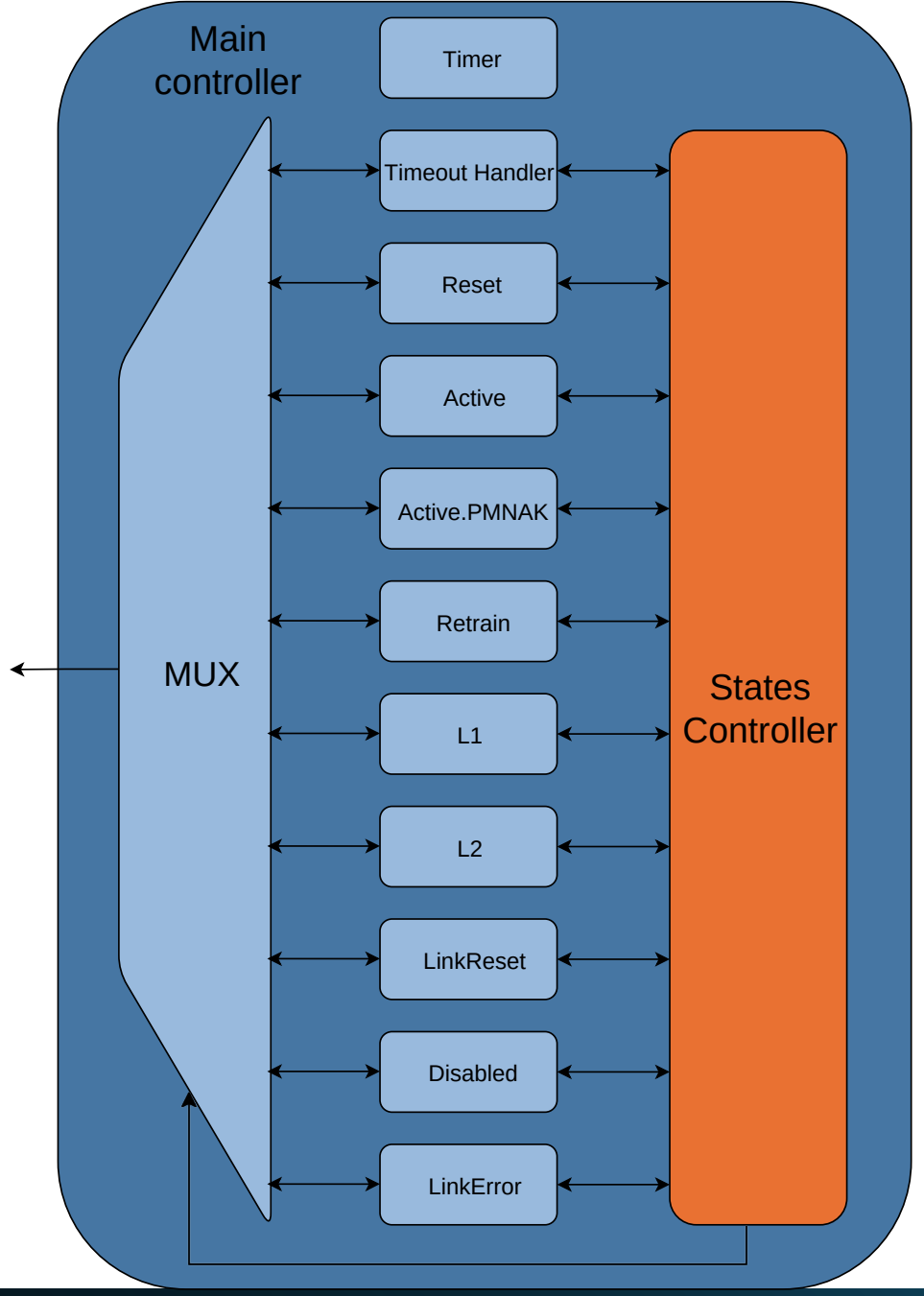


RDI SM

Main Controller

Main parts:

1. Main Controller States
2. State Controller:
   - Arbitrate per-state next_state outputs.
   - Register the current RDI state and update it on a resolved transition.
   - Drive one-hot state enables so exactly one per-state unit is active.

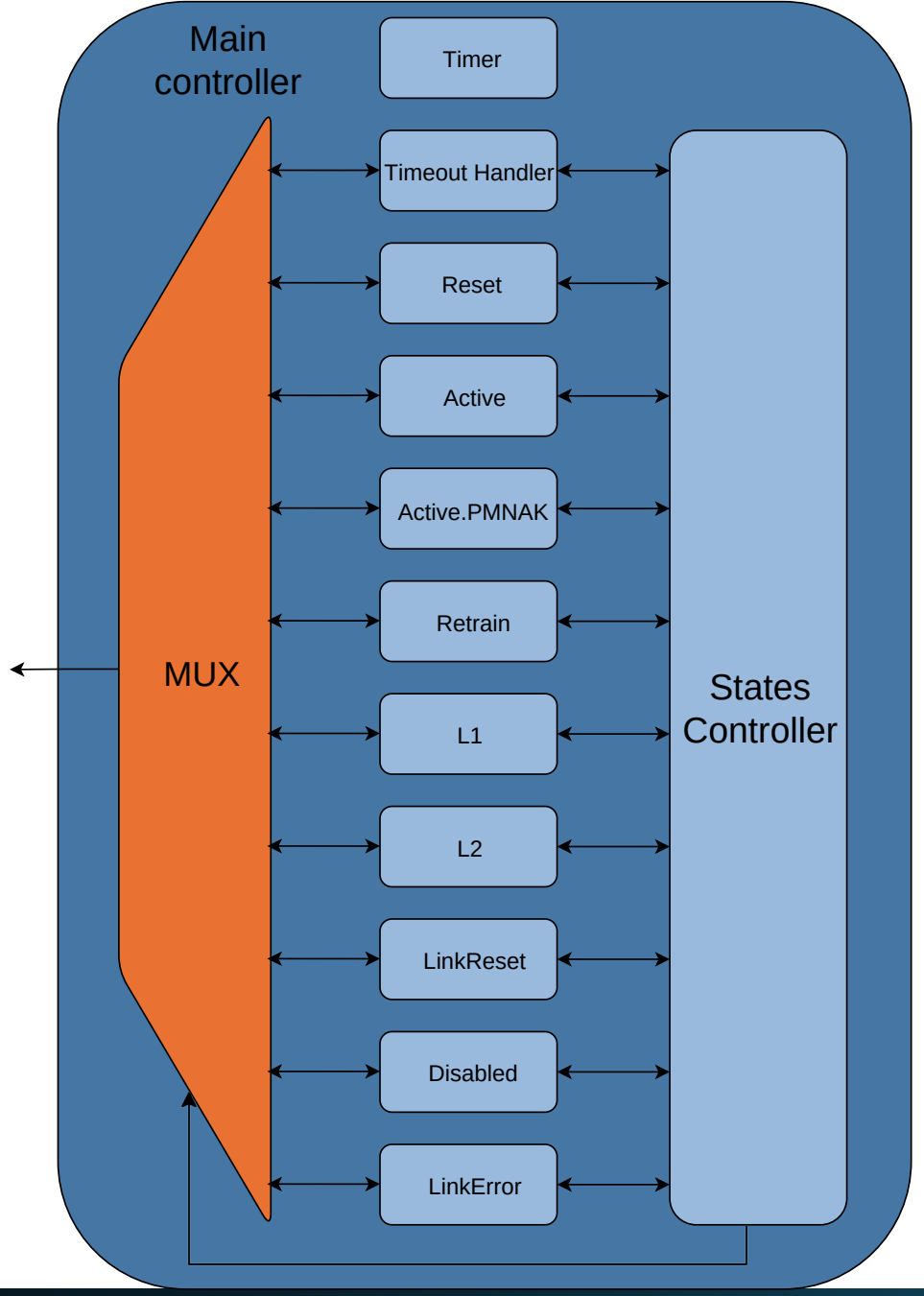


RDI SM

Main Controller

Main parts:

1. Main Controller States
2. State Controller
3. Mux:
    - State-based selection: routes the message_send output of the currently enabled state.

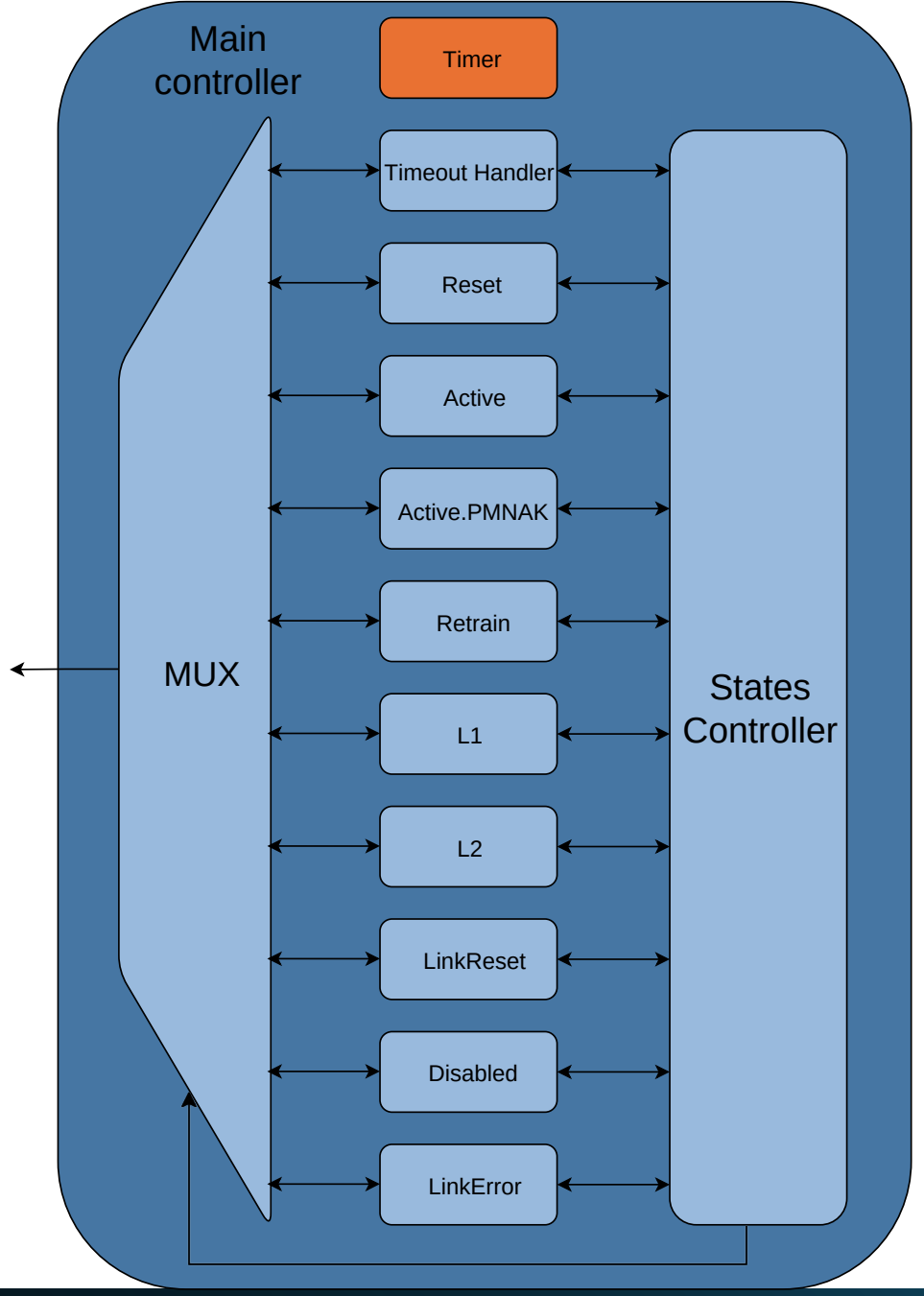


RDI SM

Main Controller

Main parts:

1. Main Controller States
2. State Controller
3. Mux
4. Timer:
    - Provides timing periods that essential for states mechanism

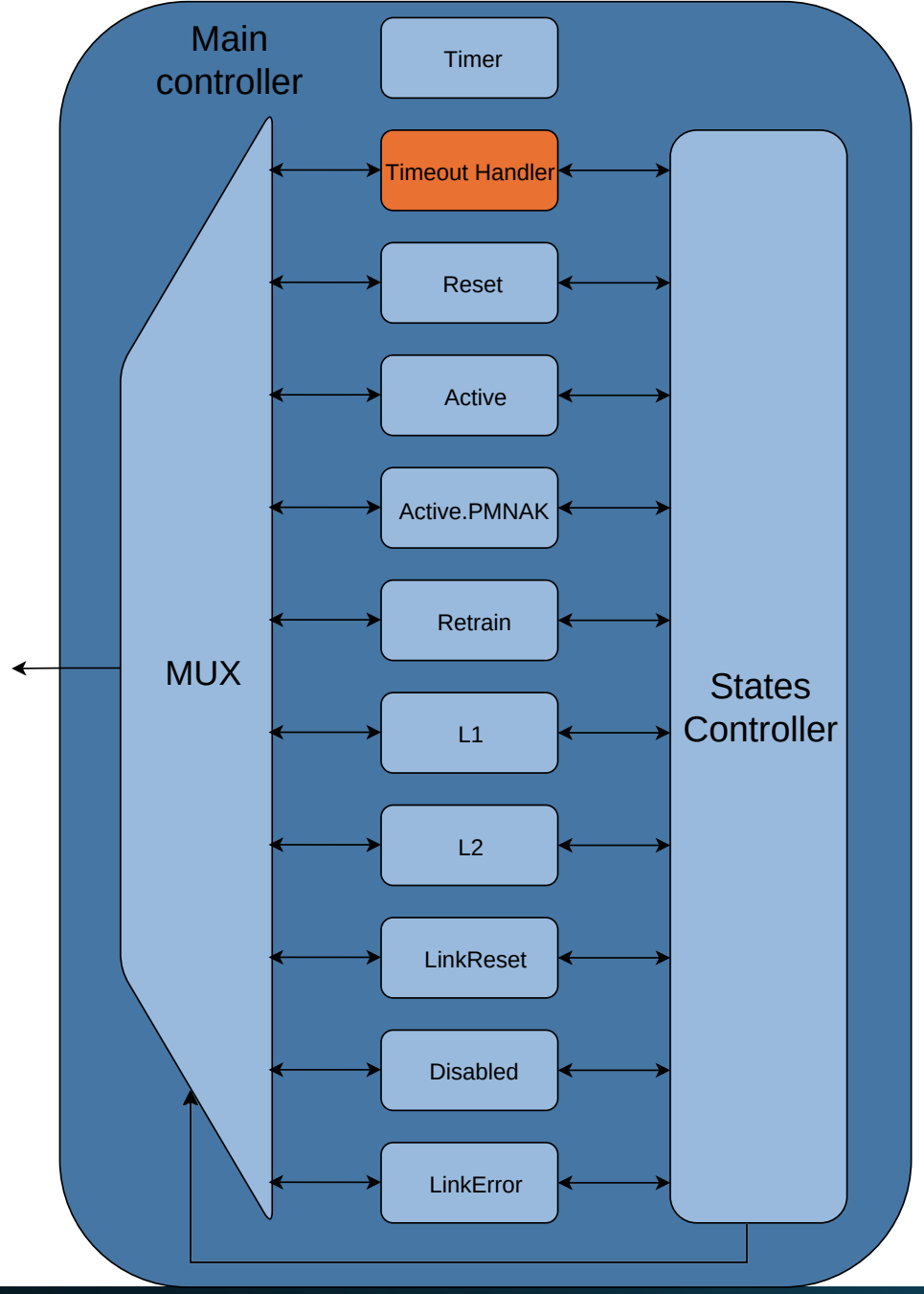


RDI SM

Main Controller

Main parts:

1. Main Controller States
2. State Controller
3. Mux
4. Timer
5. Timeout Handler:
    - Handles the timeouts for the sending requests on the link and LinkError firing on the RDI

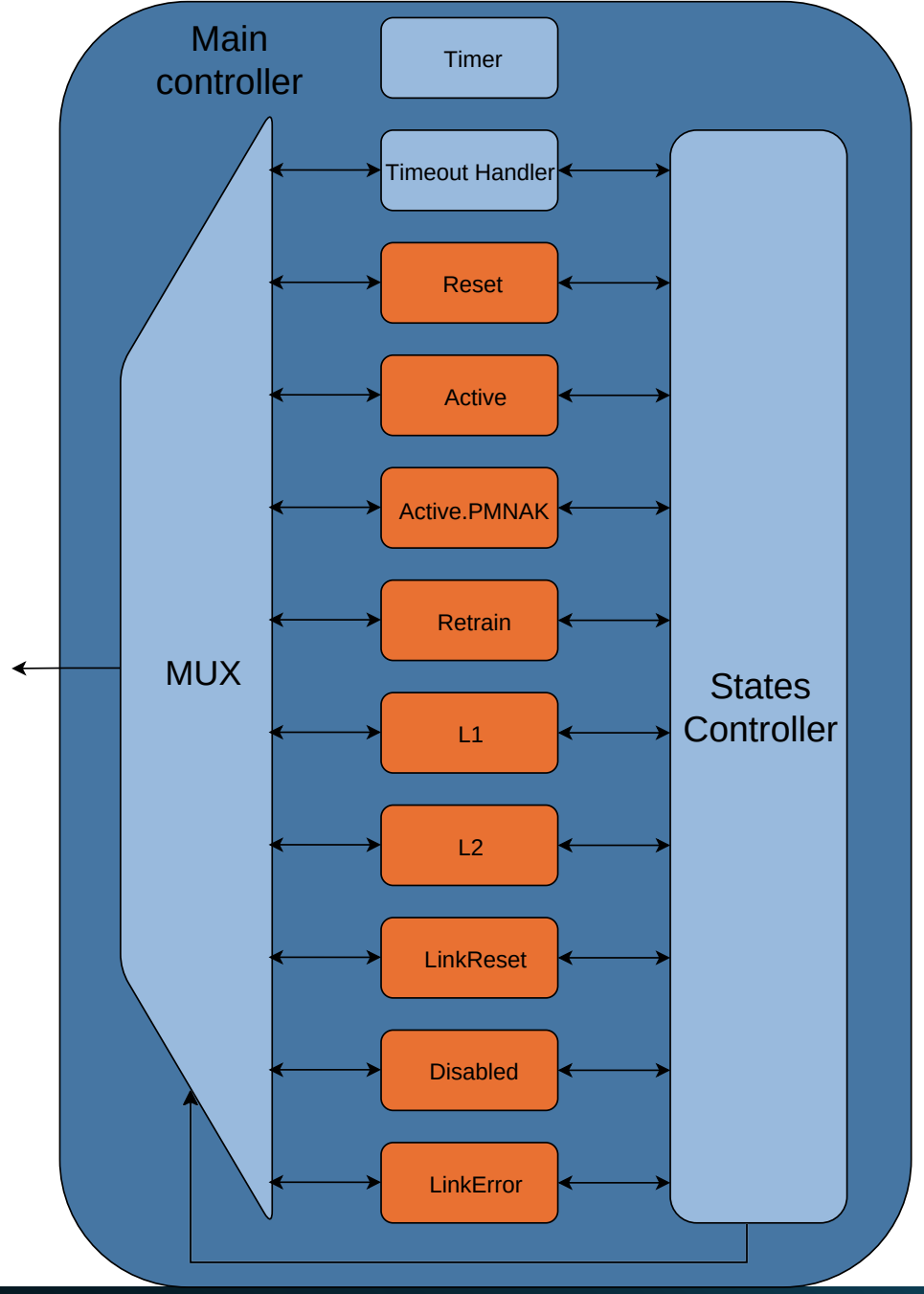


RDI SM

Main Controller

Main parts:

1. Main Controller States
    - Detect exit triggers while, received sideband message, and handshakes for a valid transition condition.
    - Execute the required sideband handshake — issue REQ/RSP (message_send) and wait for the partner's matching message, or run the Active/stall handshake, before committing.
    - Drive next_state — resolve the target state per the trigger and priority (LinkError > Disabled > LinkReset), which the main controller registers as the new rdi_state_sts.



Main Controller States

STATE 1 / 9

Reset | Active | Active.PMNAK | Retrain | L1 | L2 | LinkReset | Disabled | LinkError

Main Controller States

STATE 1 / 9

Reset | Active | Active.PMNAK | Retrain | L1 | L2 | LinkReset | Disabled | LinkError

Reset

Initial/link-down state entered on domain reset or after LinkError/LinkReset/Disabled exit

Main Controller States

STATE 2 / 9

Reset | **Active** | Active.PMNAK | Retrain | L1 | L2 | LinkReset | Disabled | LinkError

Active

Normal operational state where the mainband data path is up and traffic is permitted

Main Controller States

STATE 3 / 9

Reset | Active | Active.PMNA | Retrain | L1 | L2 | LinkReset | Disabled | LinkError

Active.PMNAK

Active variant entered when a PM (L1/L2) request cannot be granted; the layer stays operational and returns to Active on sampling an Active request (no direct arc to L1/L2 exists).

Main Controller States

STATE 4 / 9

Reset | Active | Active.PMNAK | **Retrain** | L1 | L2 | LinkReset | Disabled | LinkError

Retrain

Retraining state entered on `lp_state_req == Retrain` or an internal retrain trigger while Active

Main Controller States

STATE 5 / 9

Reset | Active | Active.PMNAK | Retrain | **L1** | L2 | LinkReset | Disabled | LinkError

L1

Power-management low-power standby (clock-gated) entered from Active

Main Controller States

STATE 6 / 9

Reset | Active | Active.PMNAK | Retrain | L1 | **L2** | LinkReset | Disabled | LinkError

L2

Deeper power-down PM state entered from Active on `lp_state_req == L2`; PM exit is performed through Reset (link fully re-brought-up).

Main Controller States

STATE 7/ 9

Reset | Active | Active.PMNAK | Retrain | L1 | L2 | **LinkReset** | Disabled | LinkError

LinkReset

Link-reset state entered from Active/Retrain/L1/L2 on `lp_state_req == LinkReset`, an internal request, or a remote sideband request;

Main Controller States

STATE 8 / 9

Reset | Active | Active.PMNAK | Retrain | L1 | L2 | LinkReset | **Disabled** | LinkError

Disabled

Link-disabled state entered on `lp_state_req == Disabled`, internal request, or remote sideband request; higher priority than LinkReset, exits to Reset

Main Controller States

STATE 9 / 9

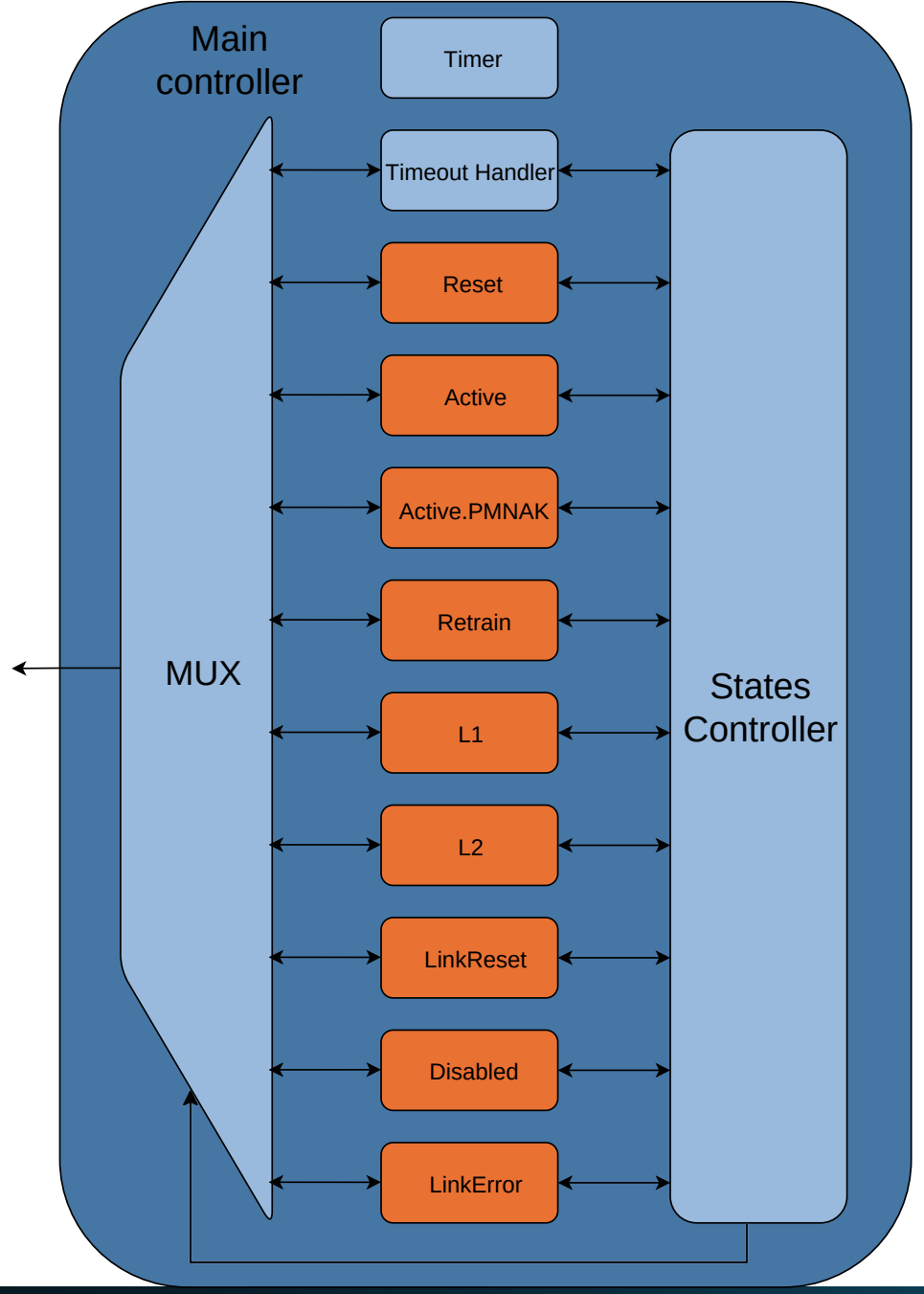
Reset | Active | Active.PMNAK | Retrain | L1 | L2 | LinkReset | Disabled | **LinkError**

LinkError

Error state entered on internal uncorrectable error, `lp_linkerror` assertion, or remote sideband request; highest transition priority

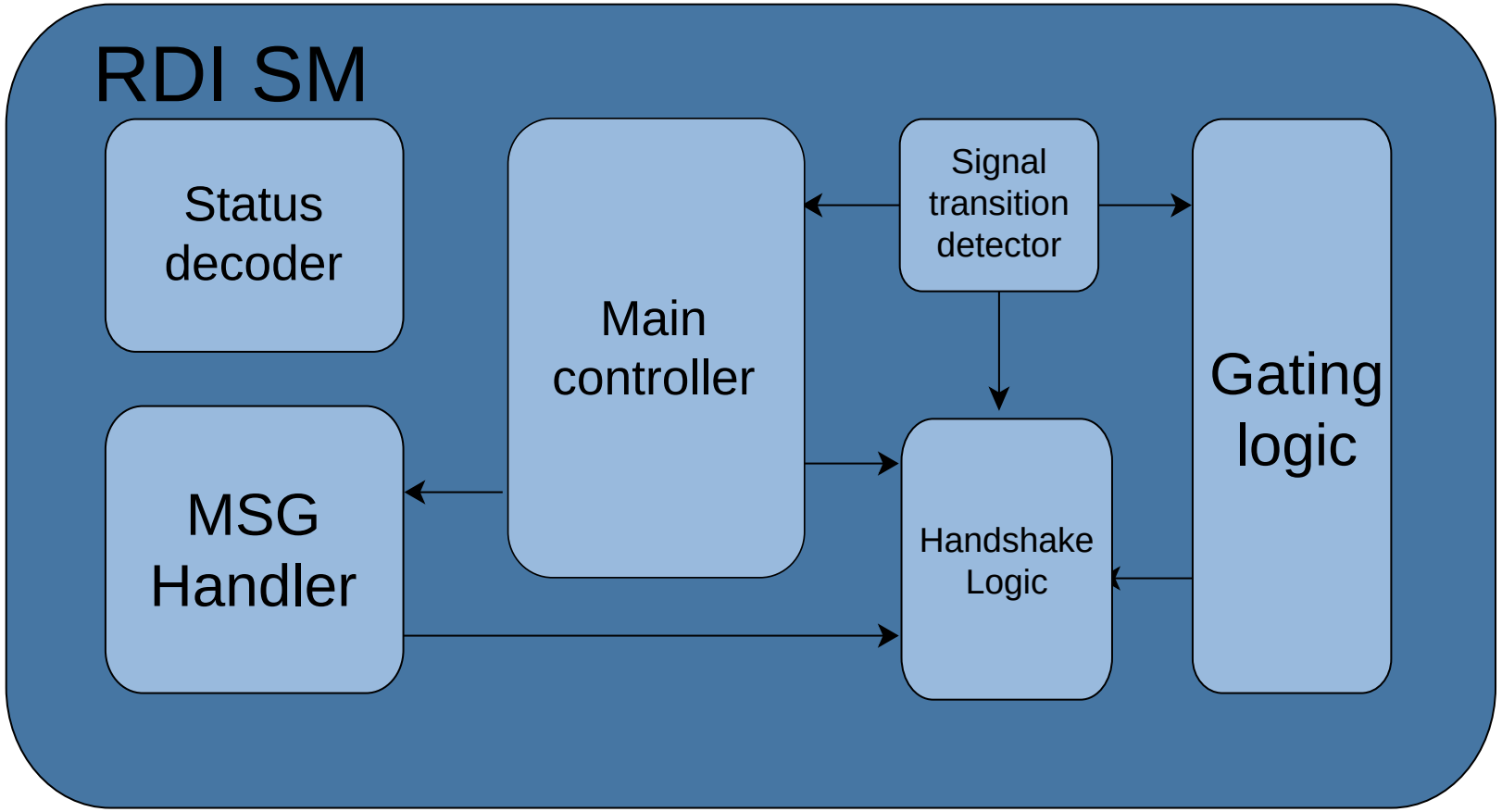
RDI SM

Main Controller



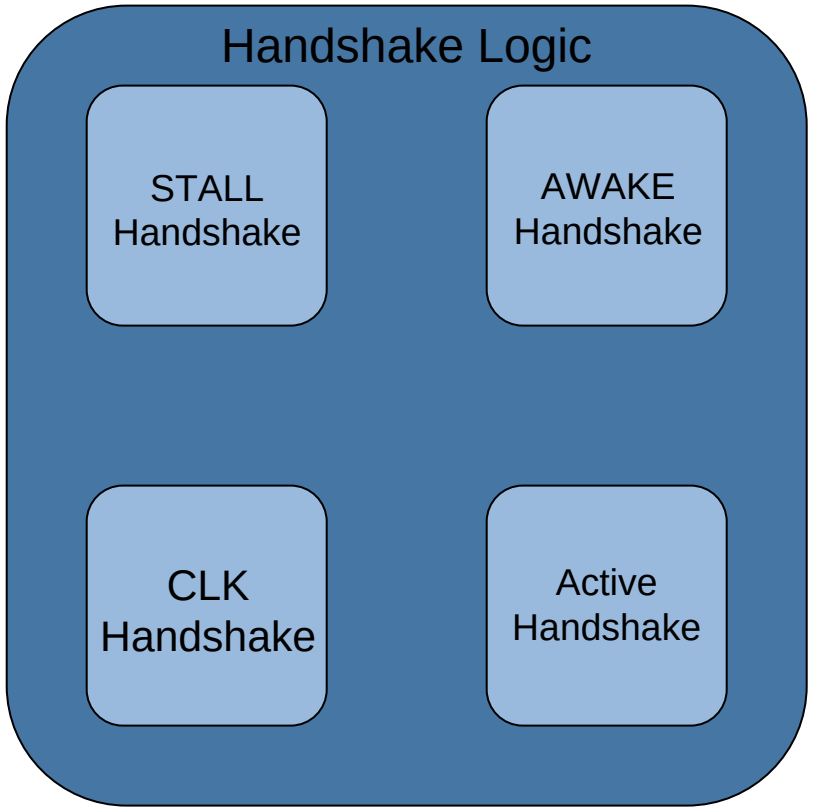
RDI SM

Link Management and Handshakes Logic



RDI SM

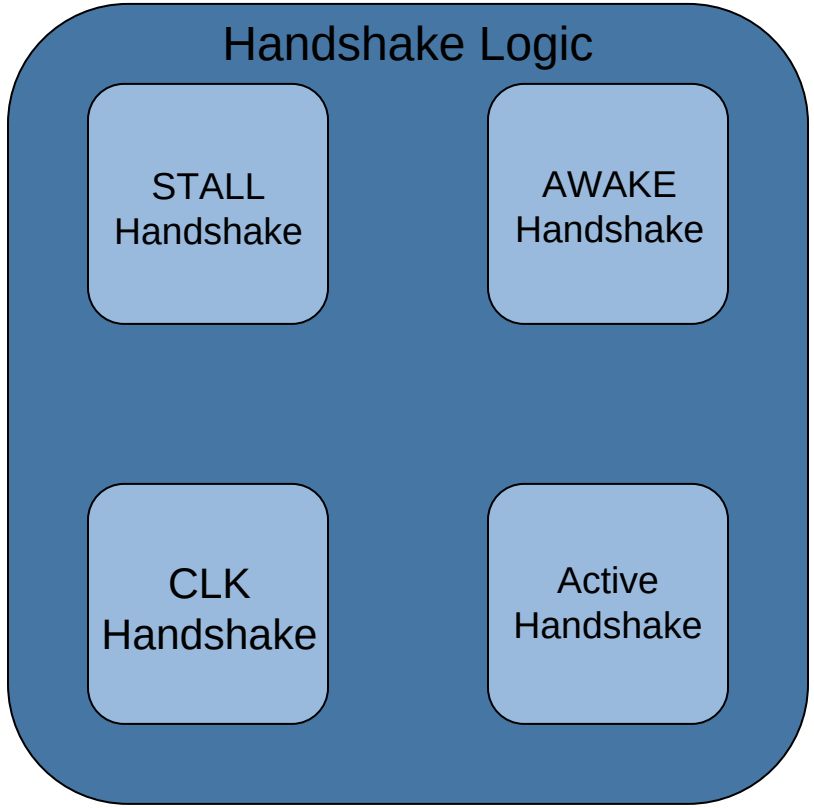
Handshake Logic



RDI SM

Handshake Logic

Main Parts:

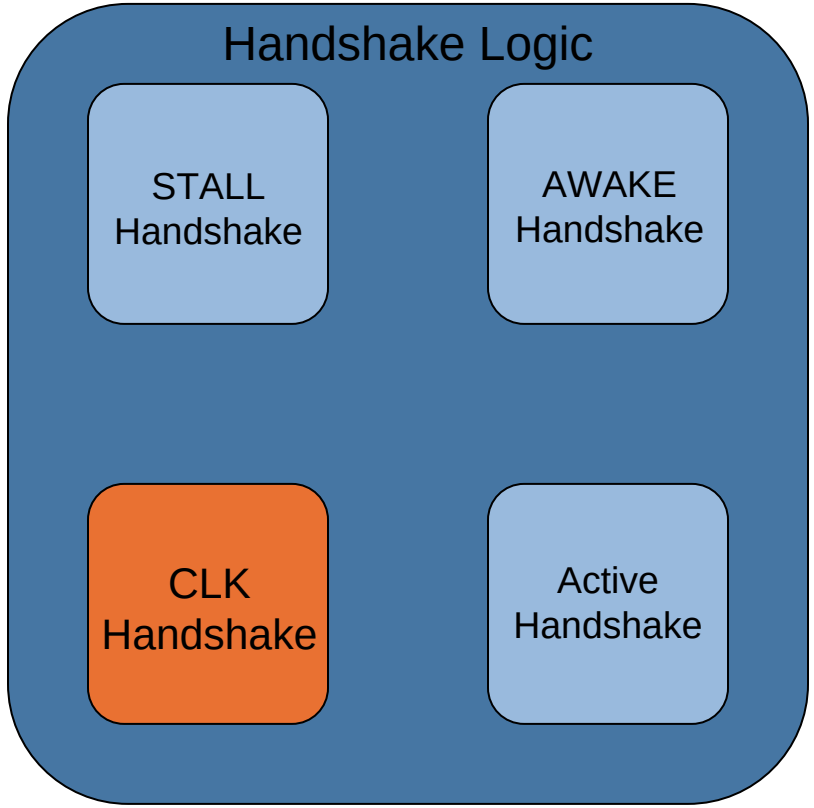


RDI SM

Handshake Logic

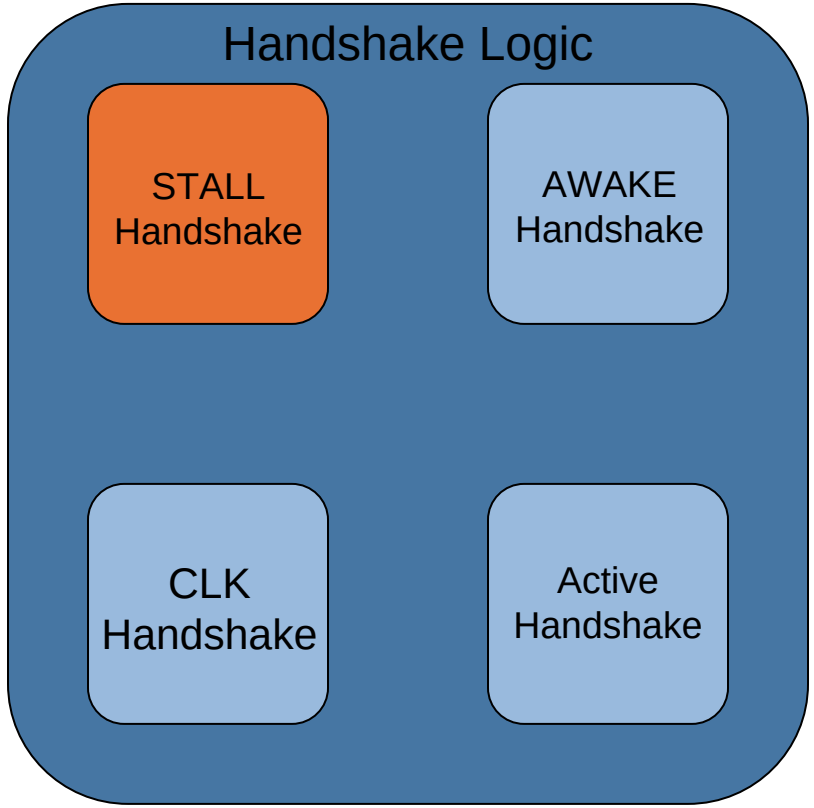
Main Parts:

1. CLK Handshake:
    - Before the PHY updates adapter-visible outputs (pl_state_sts, pl_phyinrecenter, pl_inband_pres, pl_trainerror), it drives pl_clk_req and waits for lp_clk_ack, then asserts clk_handshake_done to gate the output update.



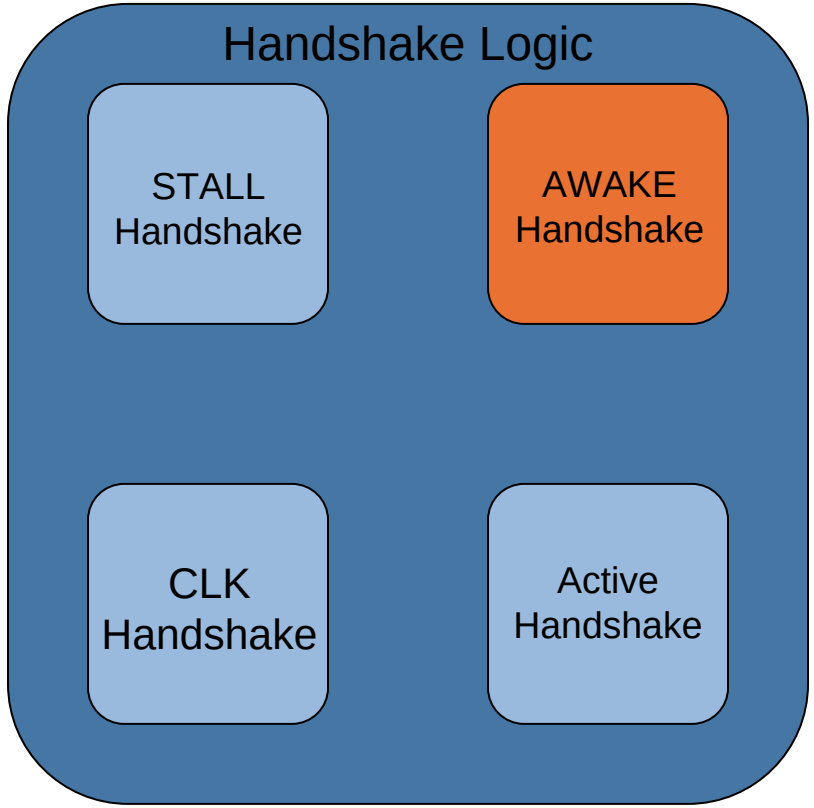
RDI SM
Handshake Logic

Main Parts:
1. CLK Handshake
2. STALL Handshake:
    - When the SM requests a datapath stall (stall_req), drives pl_stallreq, waits for lp_stallack, then asserts stall_done (latched) to quiesce the mainband datapath.



RDI SM
Handshake Logic

Main Parts:
1. CLK Handshake
2. STALL Handshake
3. AWAKE Handshake:
    - On adapter lp_wake_req, requests clock ungating and waits for ungating_done, then asserts pl_wake_ack to confirm the mainband clock is restored.

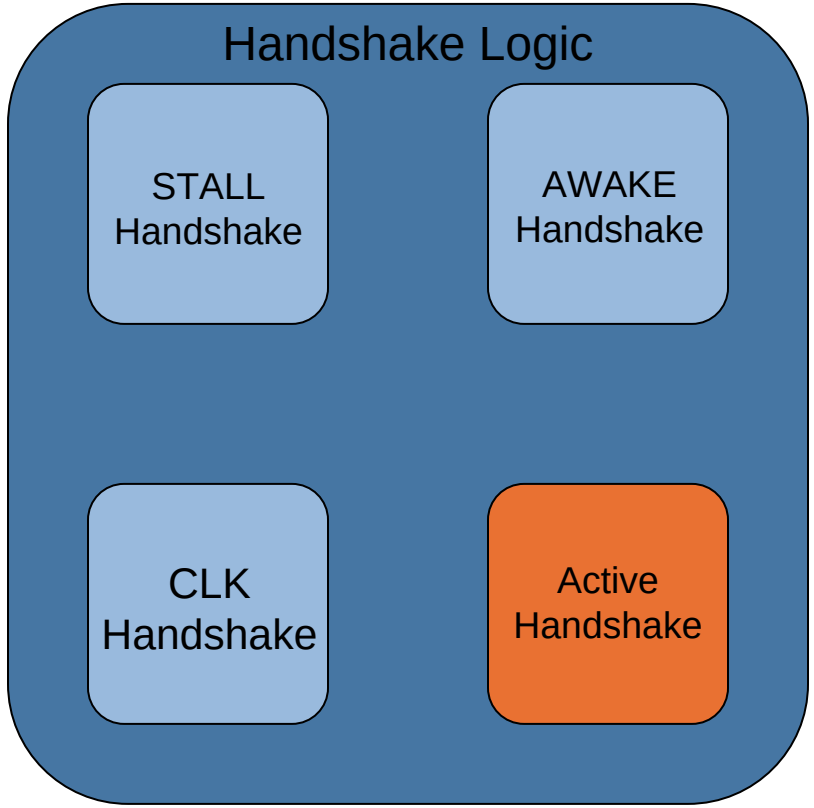


RDI SM

Handshake Logic

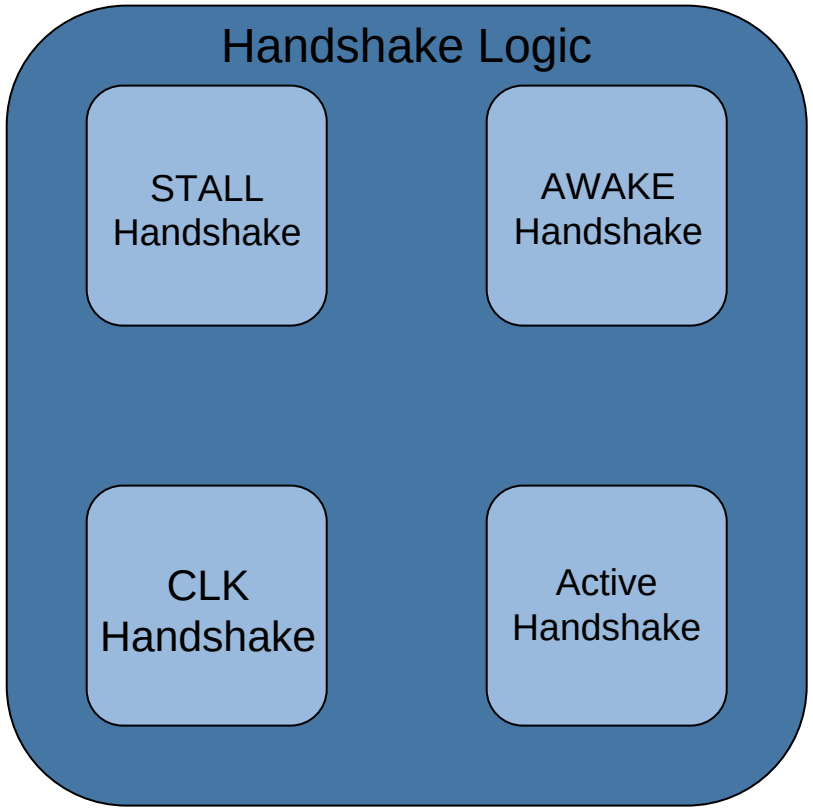
Main Parts:

1. CLK Handshake
2. STALL Handshake
3. AWAKE Handshake
4. ACTIVE Handshake:
    - Coordinates entry into Active by sending RDI_ACTIVE_REQ (qualified by inband_pres/pm_exit) and waiting for RDI_ACTIVE_RSP, then asserts Active_handshake_done.



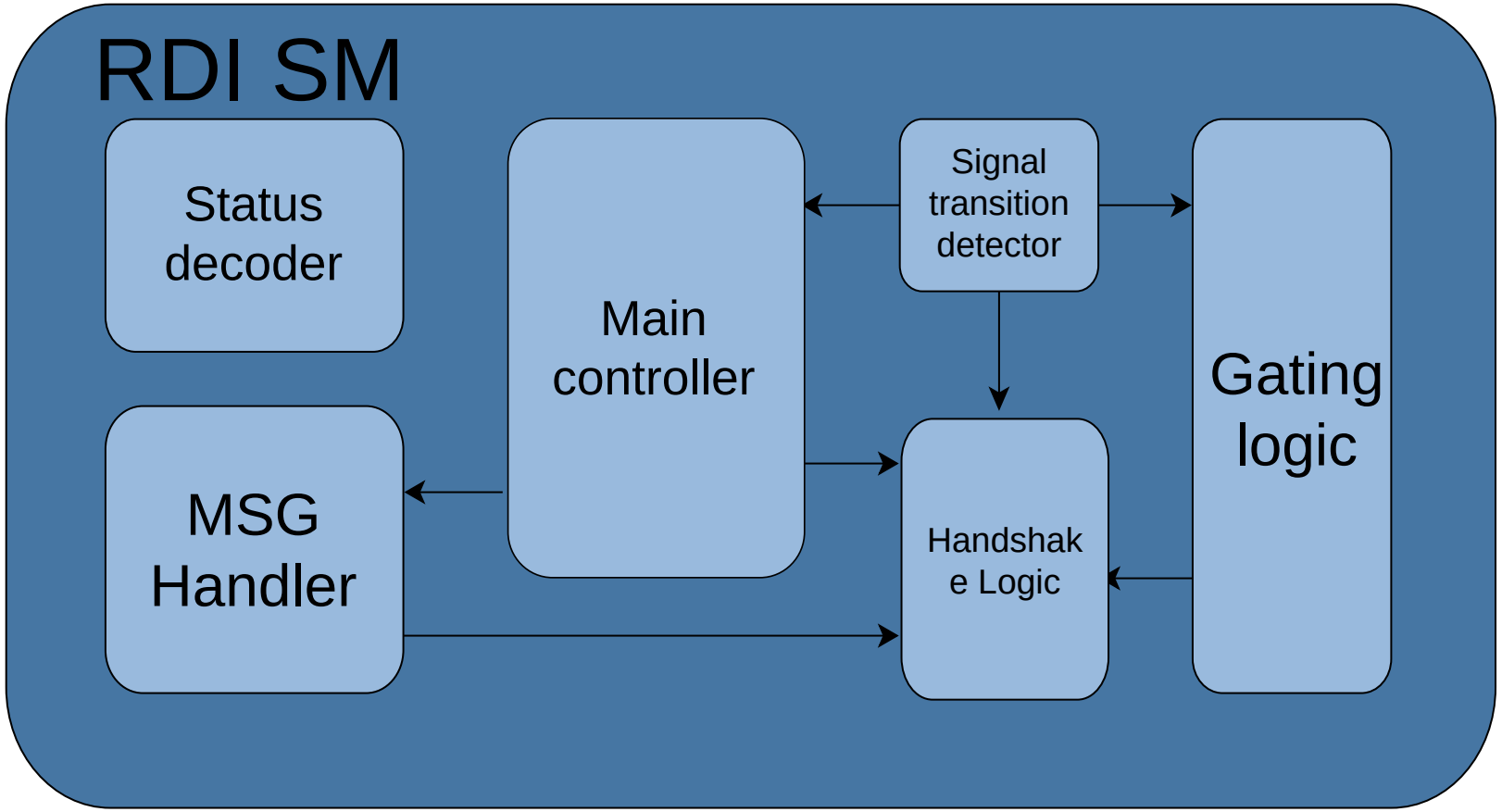
RDI SM

Handshake Logic



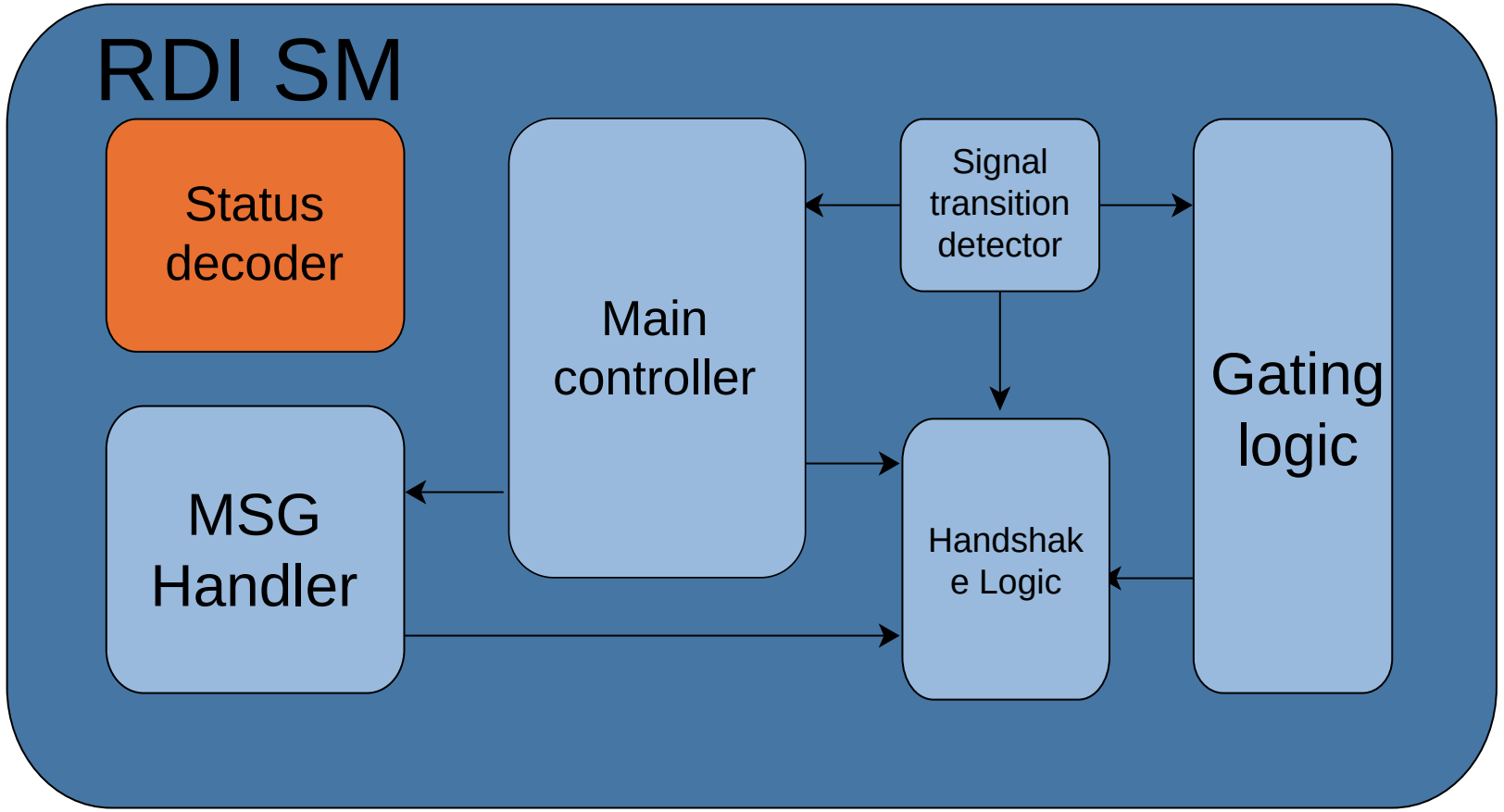
RDI SM

Link Management and Handshakes Logic



RDI SM
Status Decoder

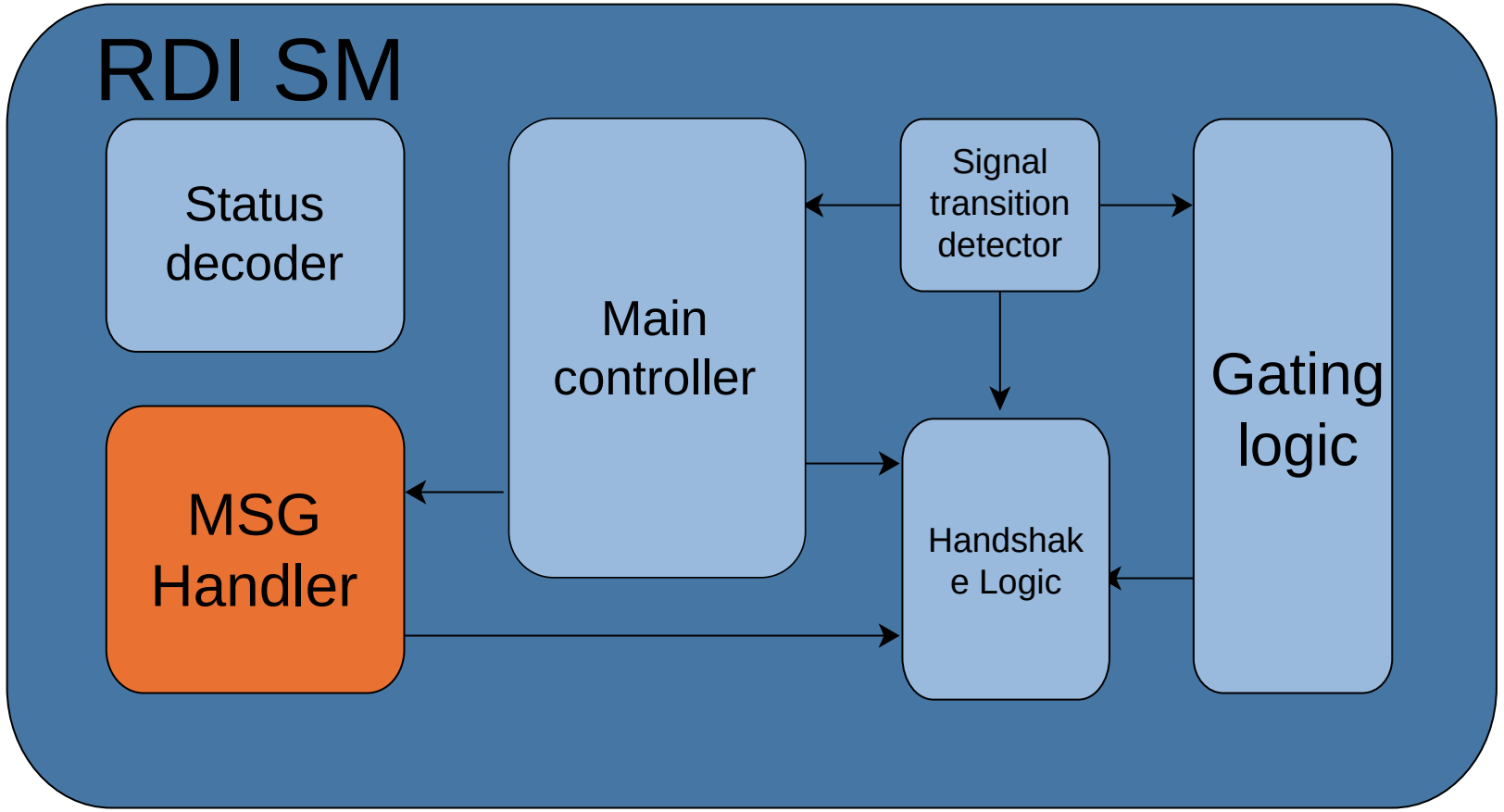
Maps Reg_File Link Capability/Status fields into pl_max_speedmode, pl_lnk_cfg, and pl_speedmode for the RDI adapter interface.



RDI SM

Messeege Handler

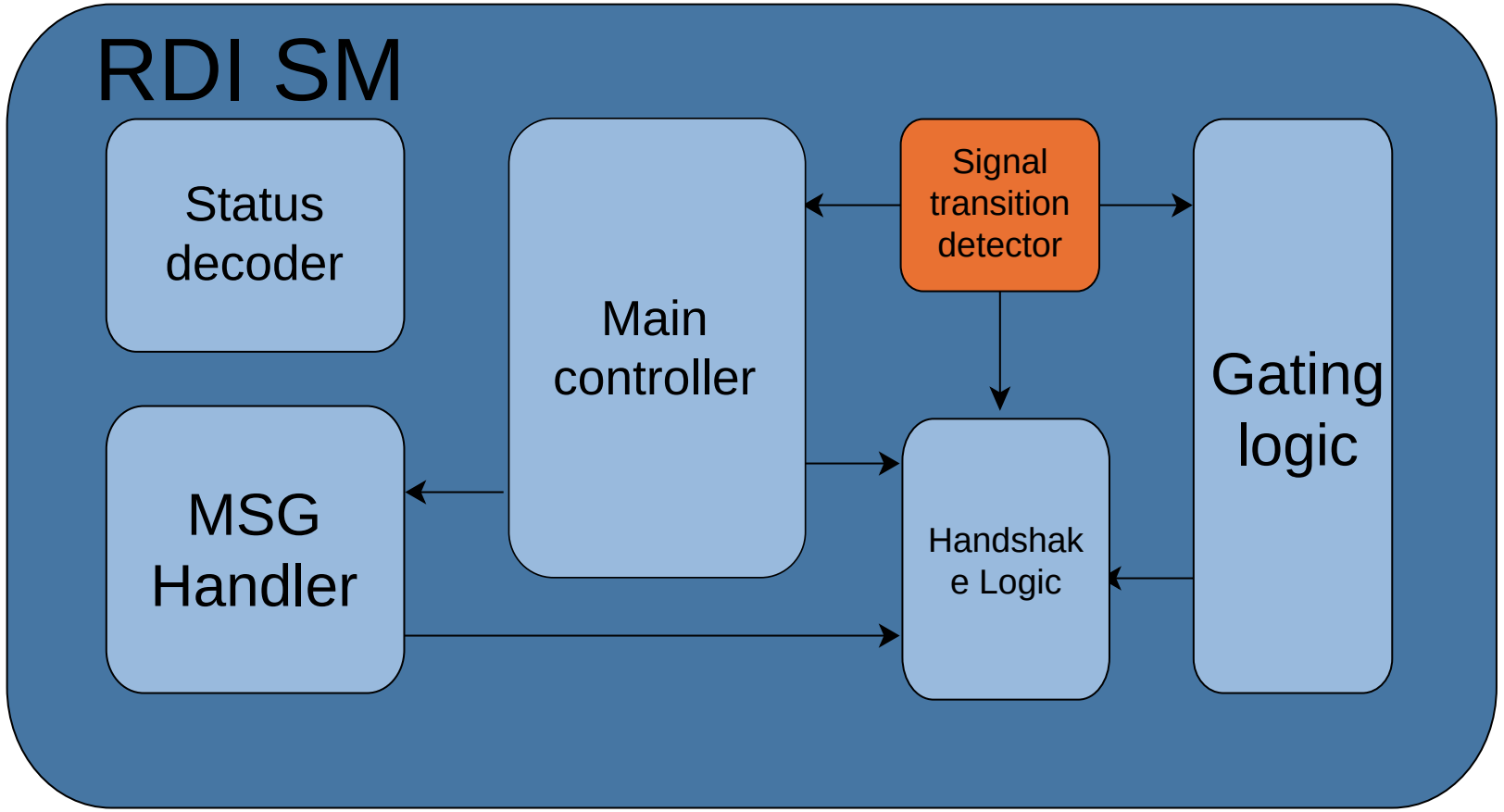
Serializes outgoing/incoming RDI sideband link-management messages with priority arbitration and a one-entry receive buffer.



RDI SM

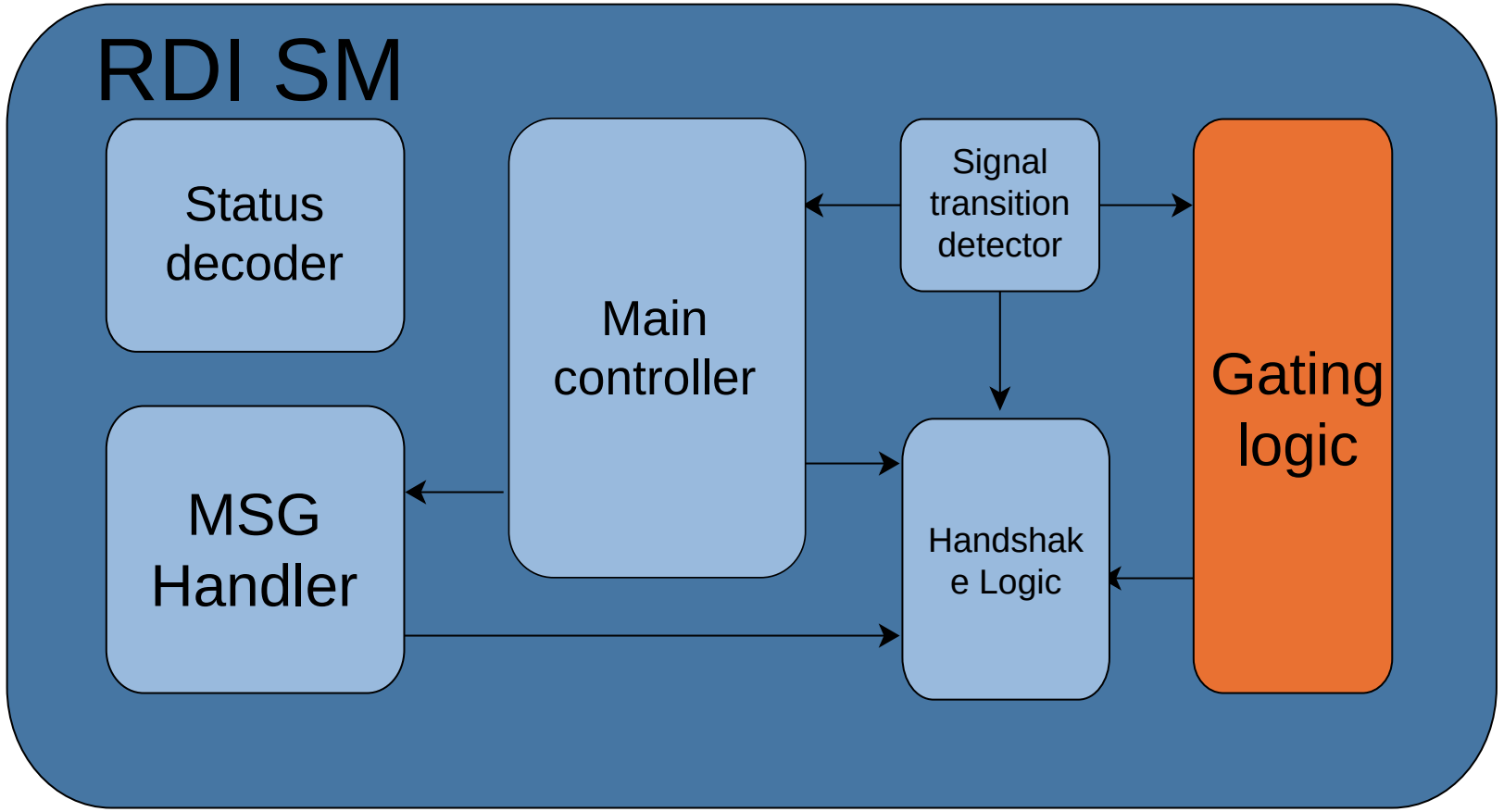
Signal Transition Detector

2-state that forces pl_state_sts, pl_phyinrecenter, pl_inband_pres, and pl_trainerror updates to the Adapter through the CLK handshake to guarantee glitch-free, edge-aligned delivery.



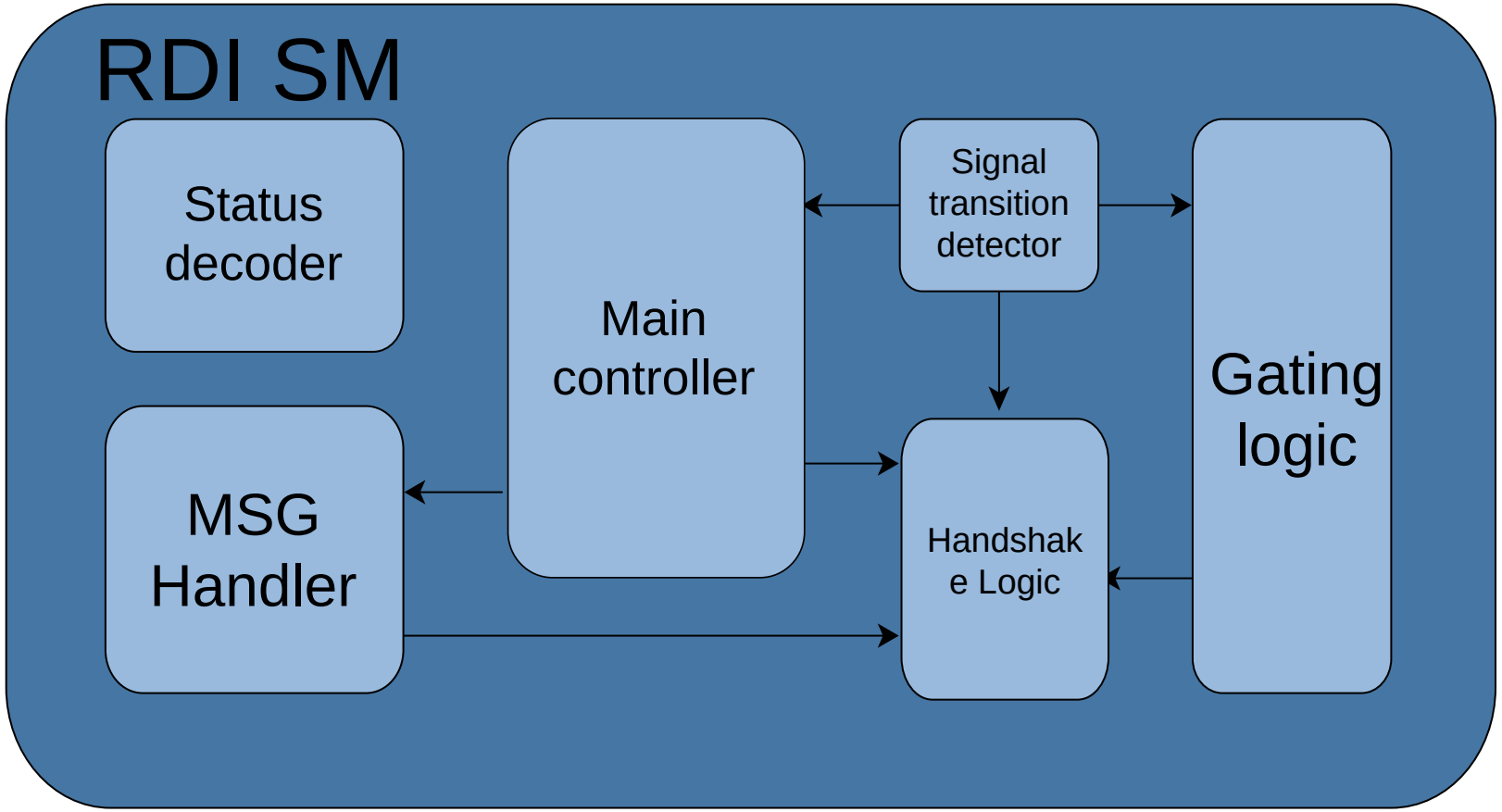
RDI SM
Gating Logic

State-driven block that gates/ungates the mainband TX clock (lclk_g) based on pl_state_sts and training-start conditions, requesting ungating via handshake with the Adapter.

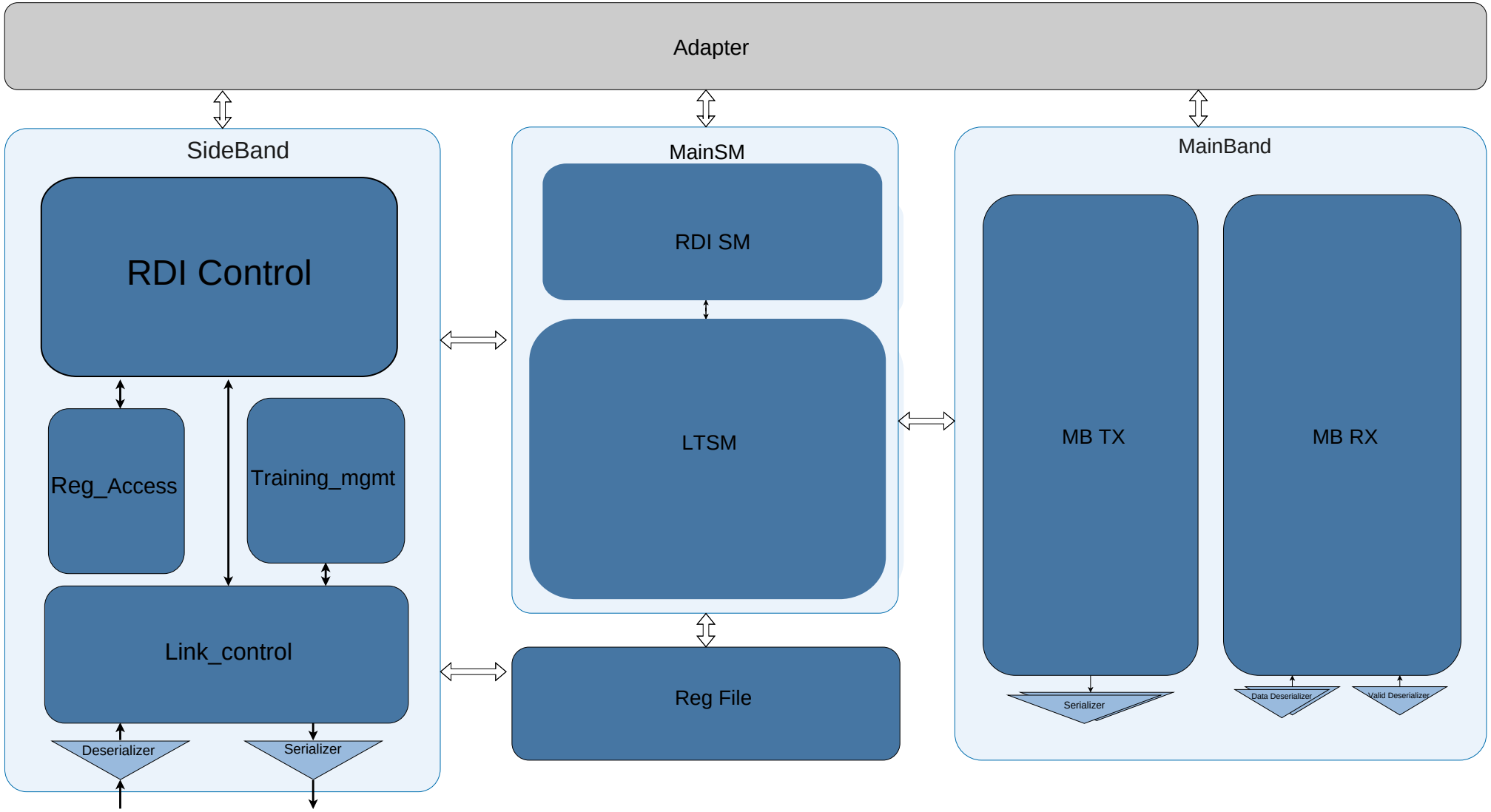


RDI SM

Link Management and Handshakes Logic



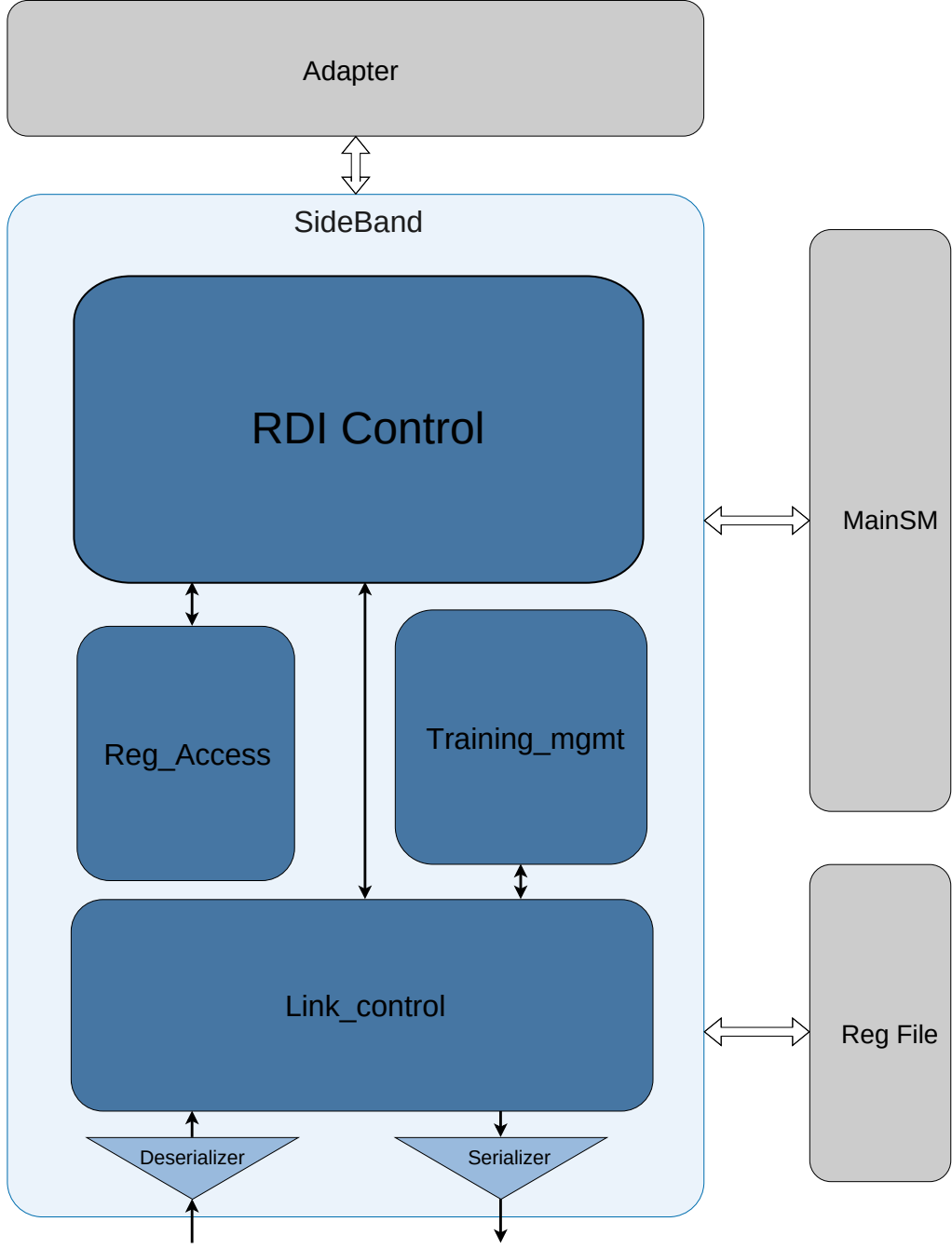
CIe PHY Architecture



SideBand

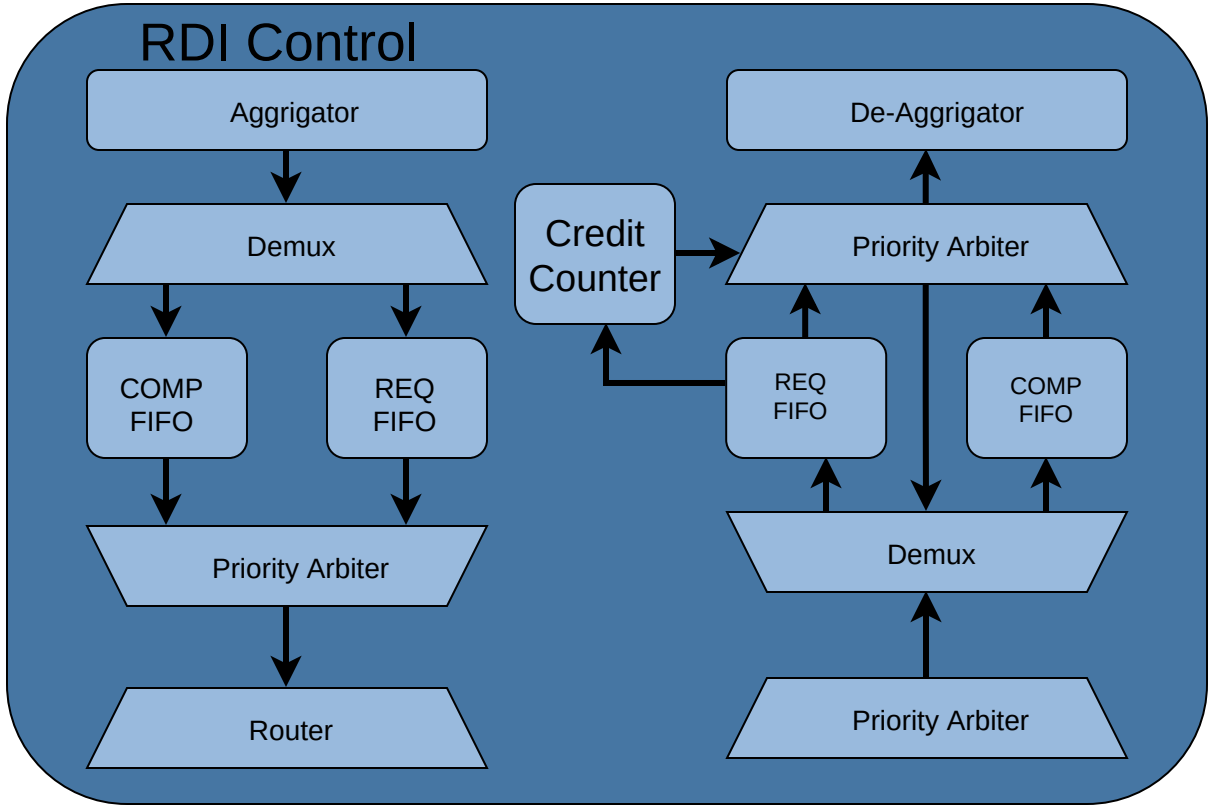
Always-On Control Plane

- Always-On Control Plane: Low-speed channel to initialize the Main-Band.
- Independent Interface: Forwarded-clock serial link separate from the high-speed data path.
- Modular Architecture: Four integrated blocks (RDI Control, Training Mgmt, Reg Access, Link Control).



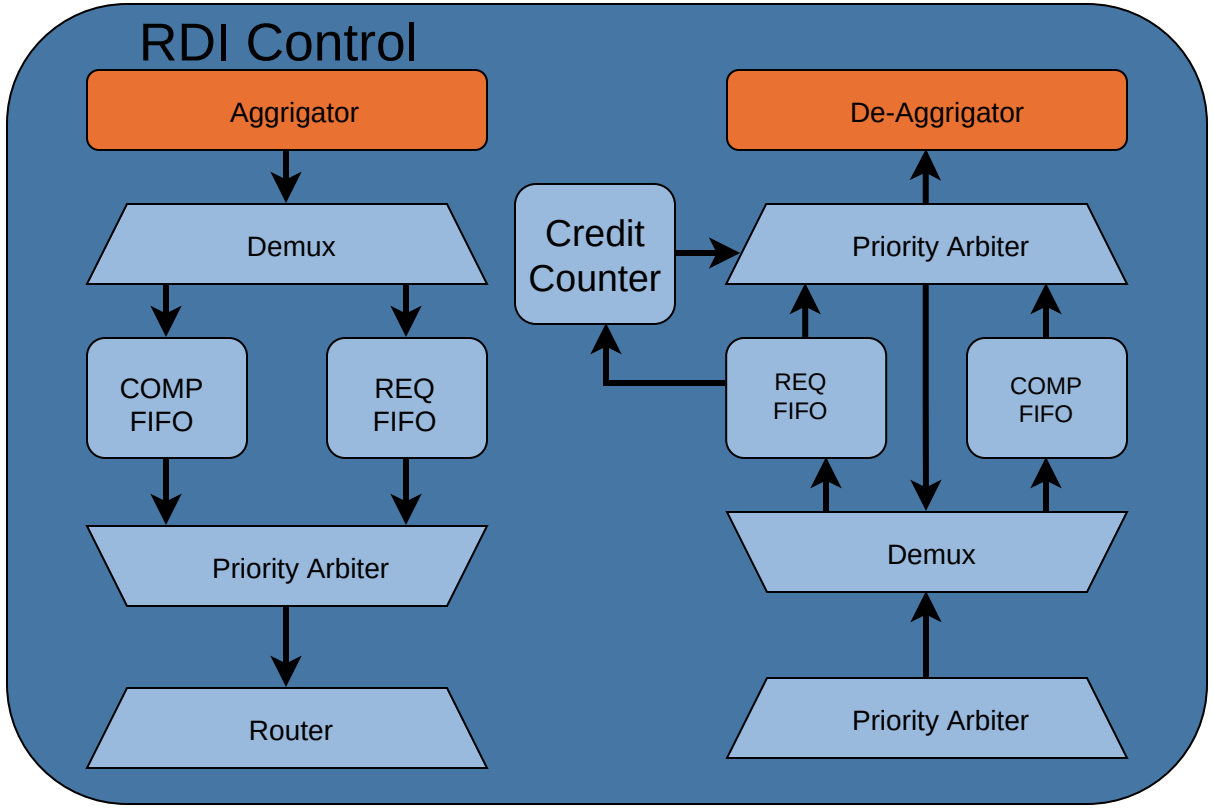
SideBand

RDI Control



SideBand
RDI Control

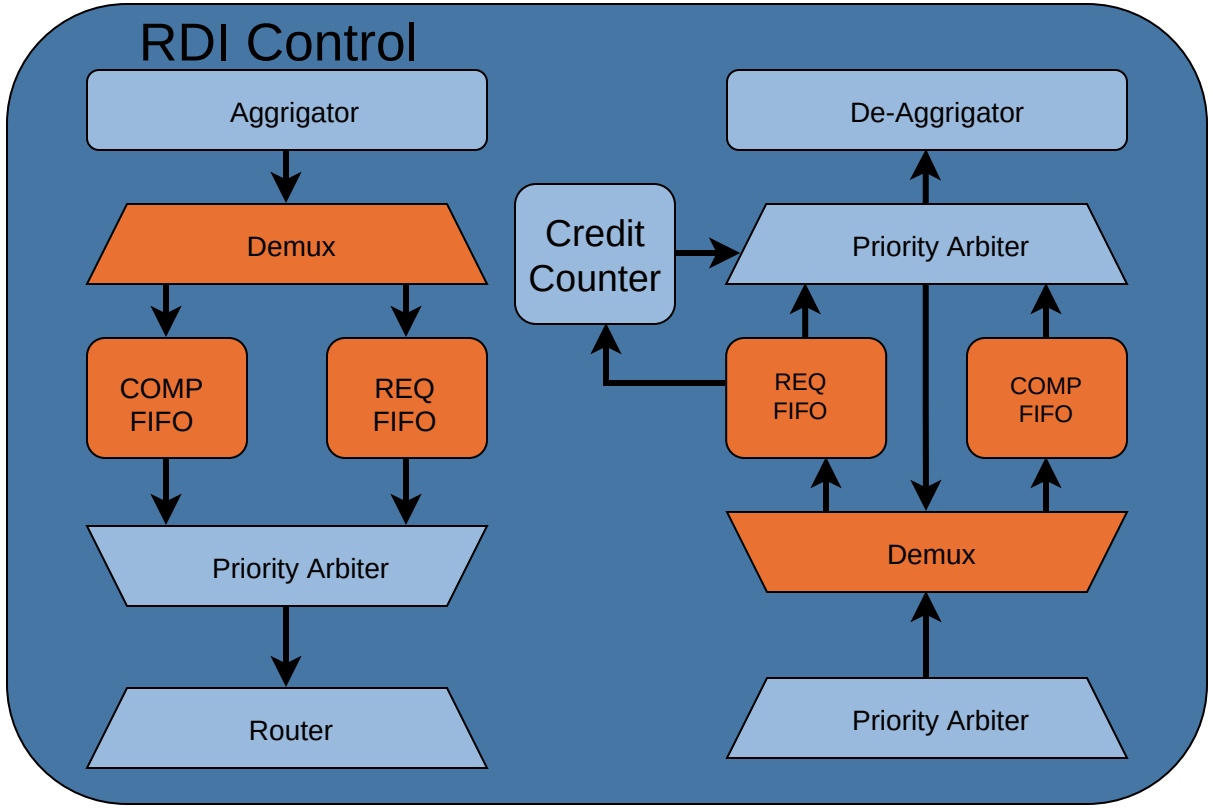
- Aggregation / De-aggregation: Converts between 32-bit adapter chunks and 128-bit packets.



SideBand

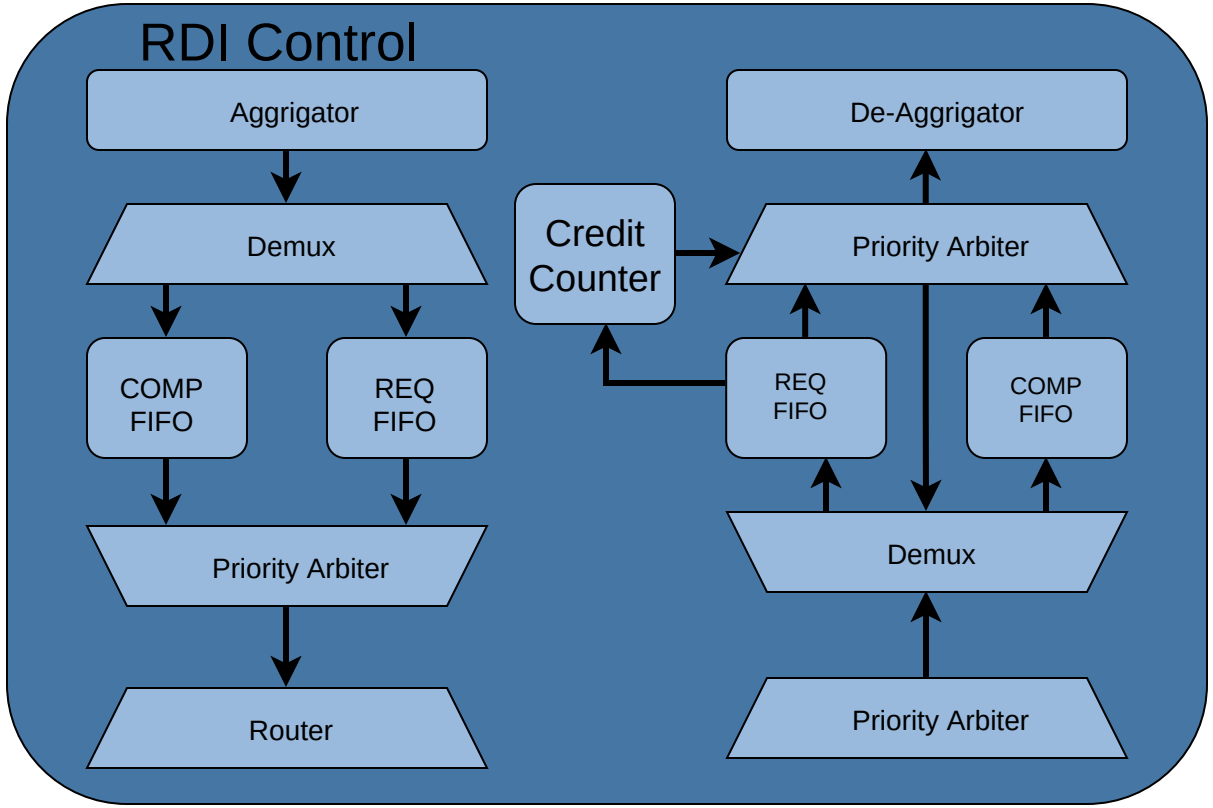
RDI Control

- Traffic Classification: Separates register-access Requests and Completions.



SideBand
RDI Control

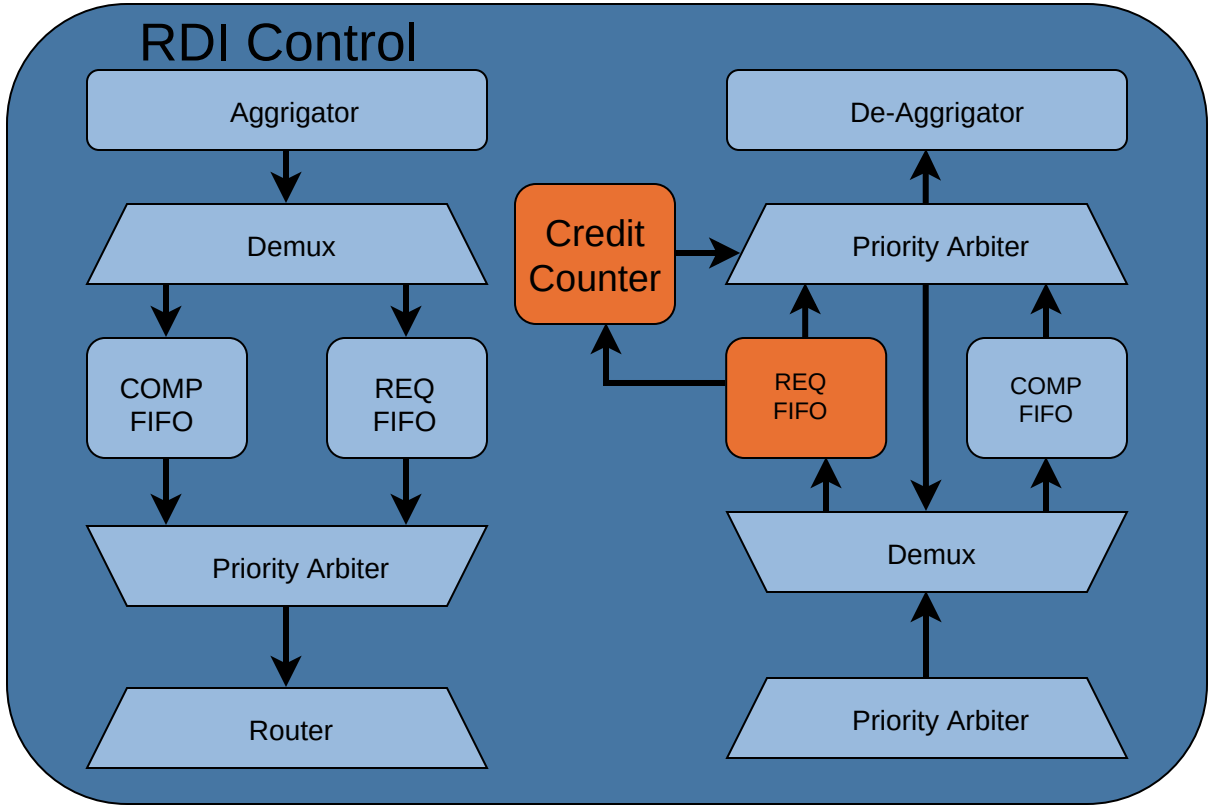
- Congestion & Deadlock Prevention:



SideBand

RDI Control

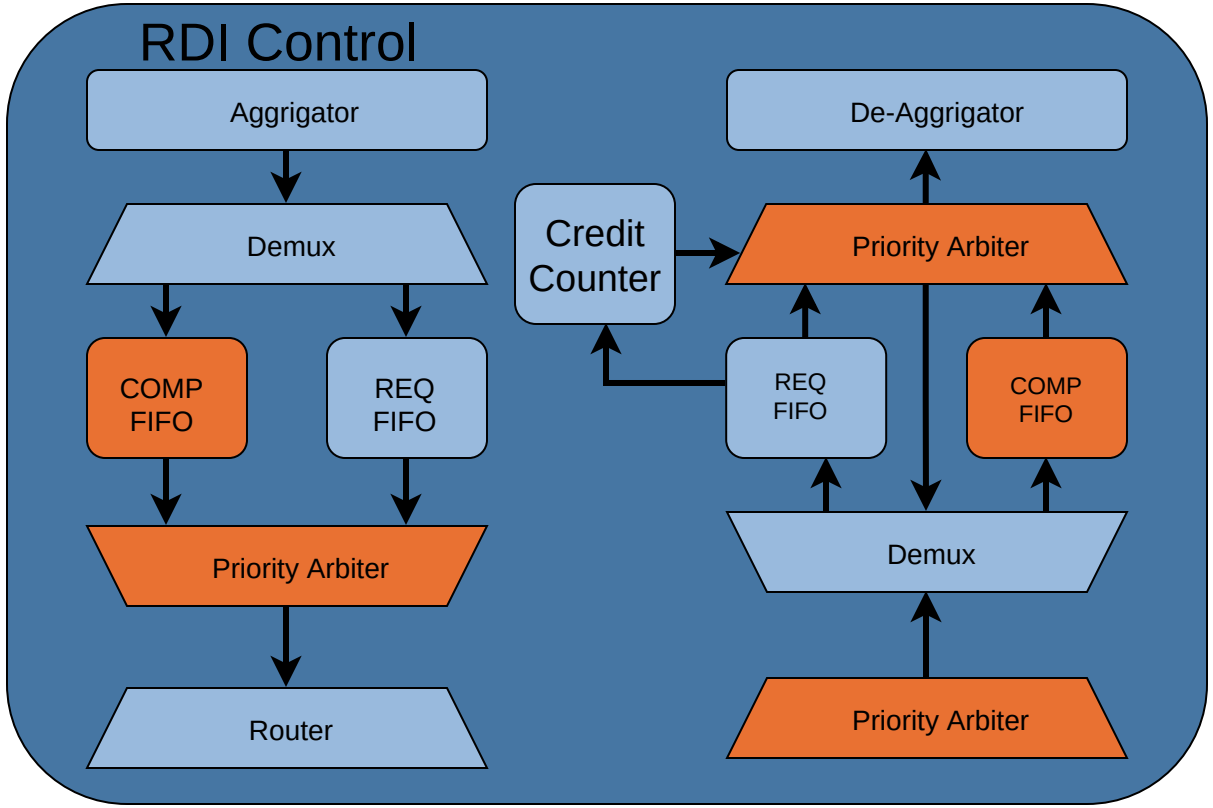
- Congestion & Deadlock Prevention:
  - Credit-based flow control for Requests.



SideBand

RDI Control

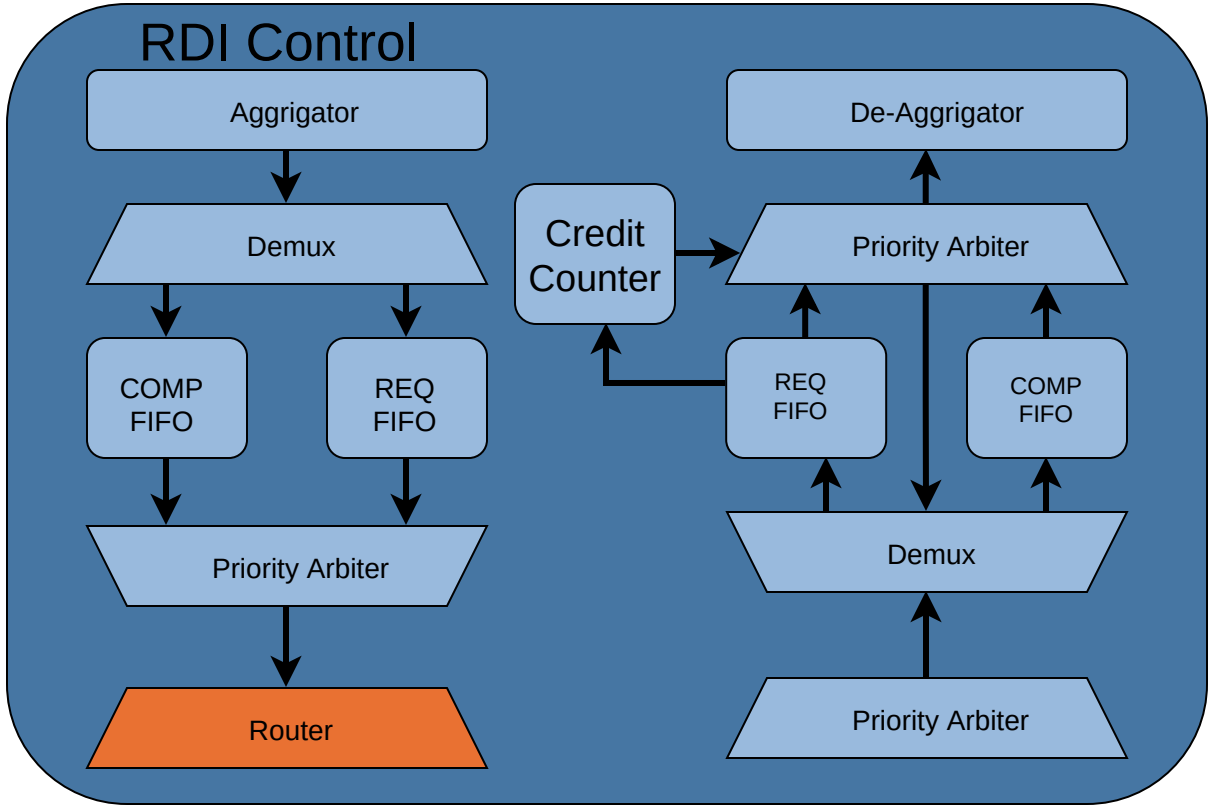
- Congestion & Deadlock Prevention:
  - Credit-based flow control for Requests.
  - Priority arbitration to ensure unconditional sinking of Completions.



SideBand

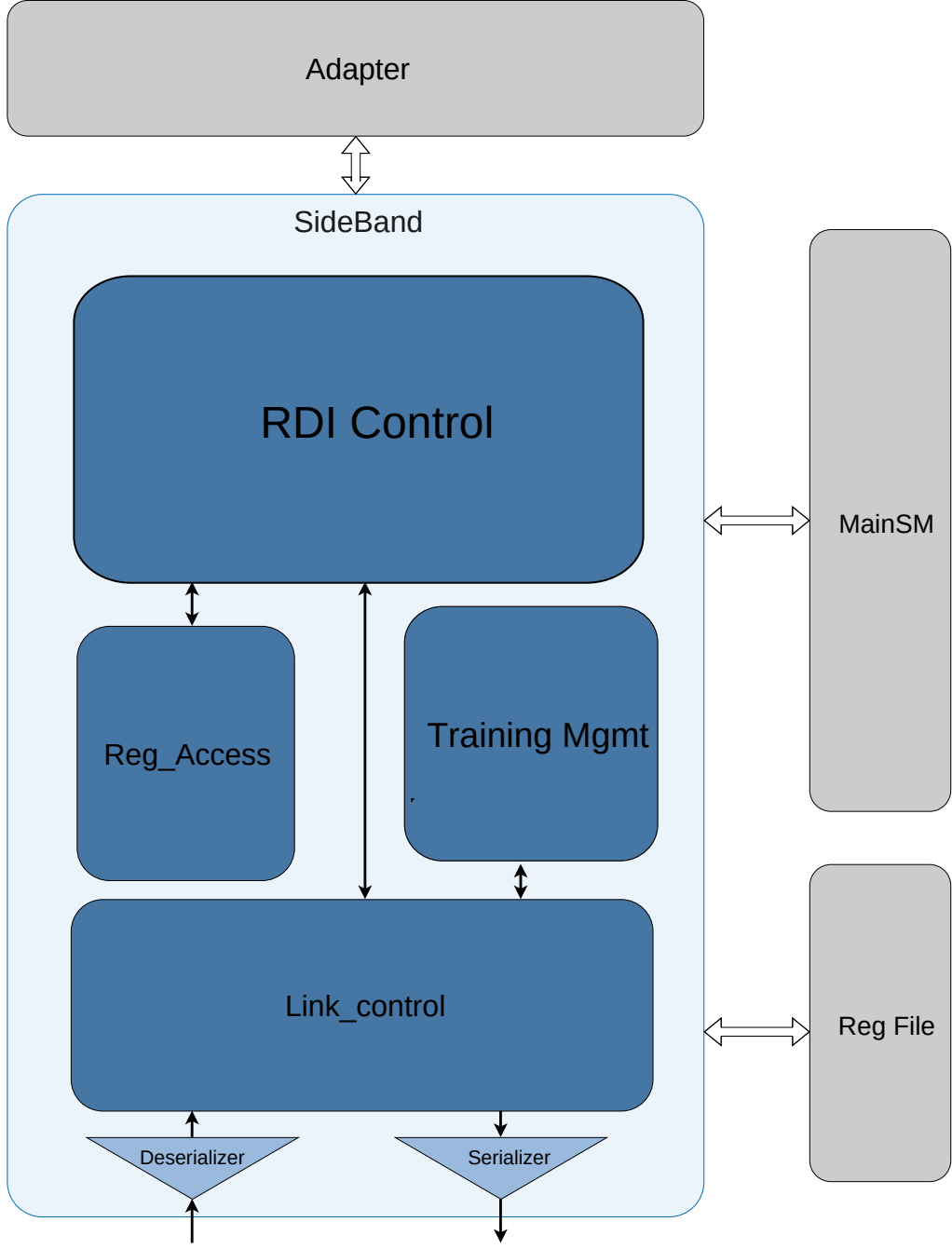
RDI Control

- Downstream Router: Directs traffic locally (Reg Access) or externally (Link Control).



SideBand

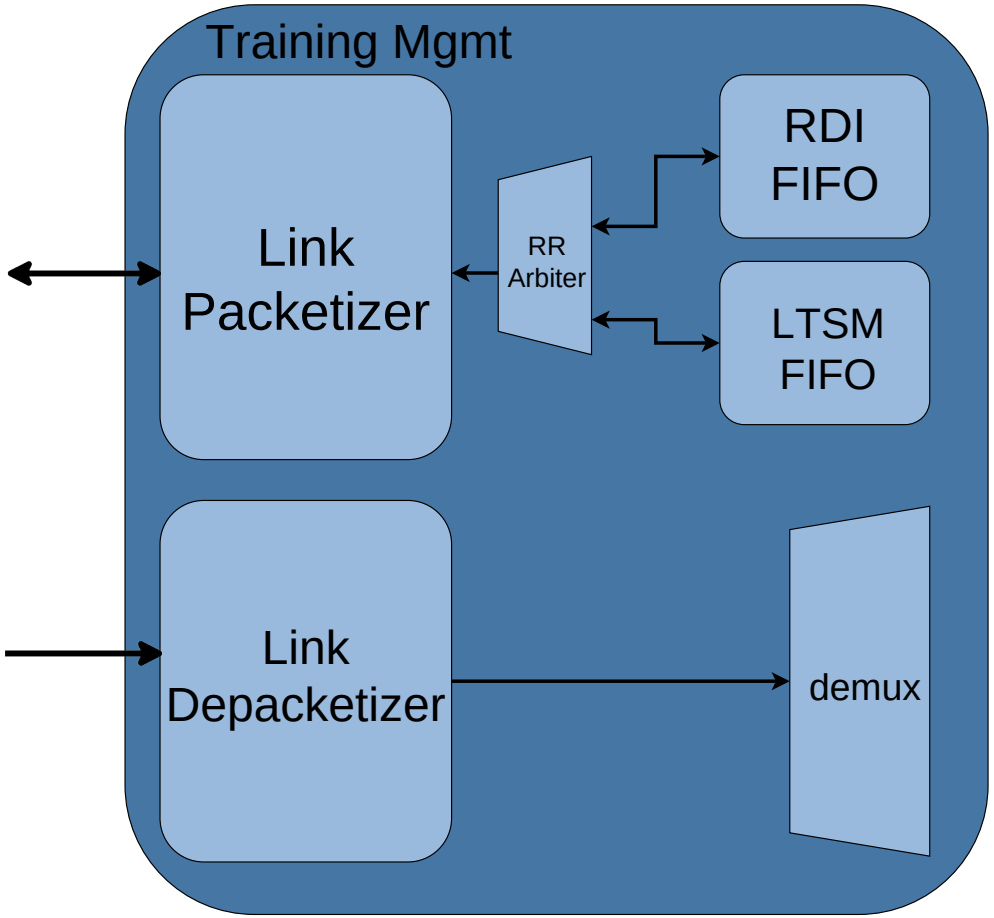
Always-On Control Plane



SideBand

Training Management

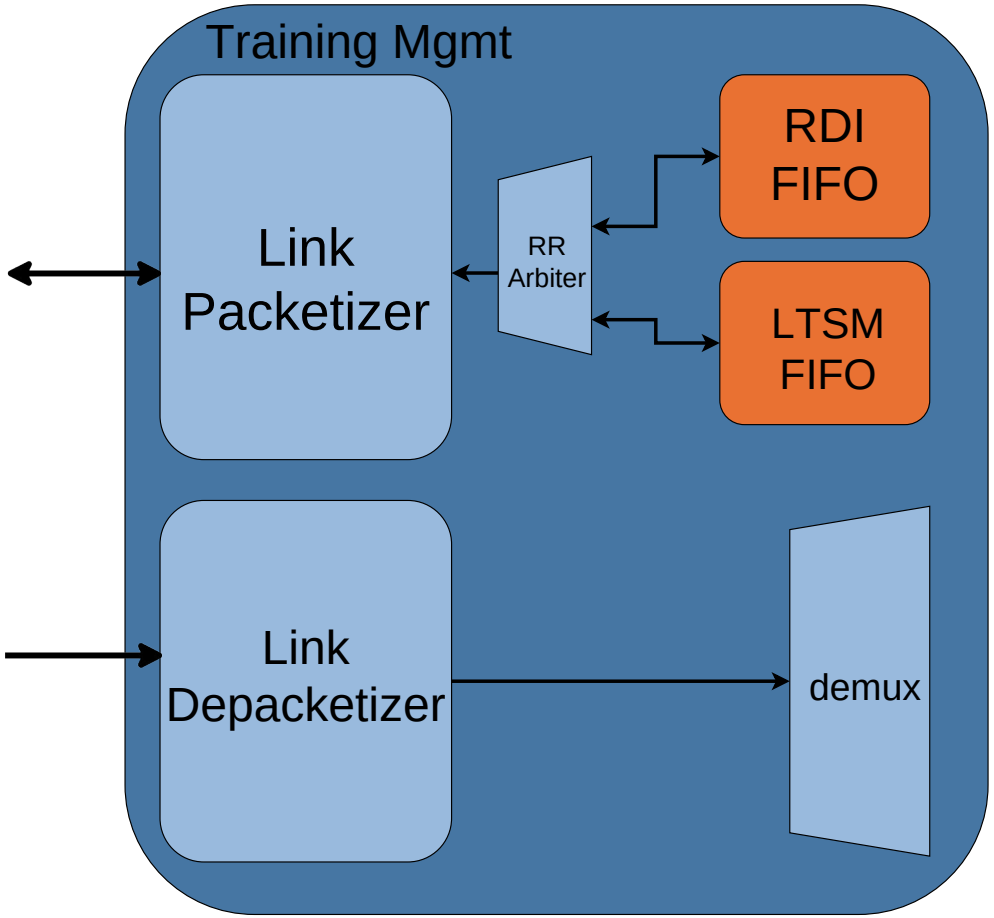
- LTSM & RDI Transport: Bridges Main SM control signals to the Sideband.



SideBand

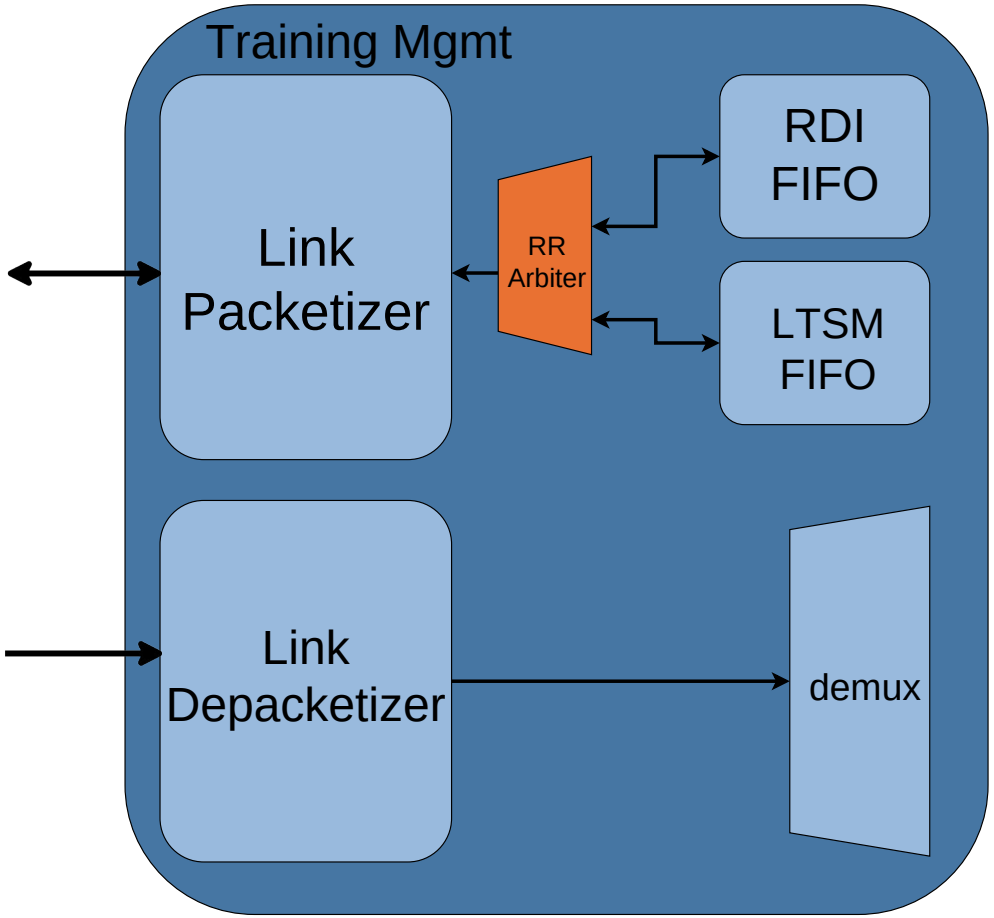
Training Management

- Metastability Protection: Asynchronous FIFOs handle Clock Domain Crossing (CDC).



SideBand
Training Management

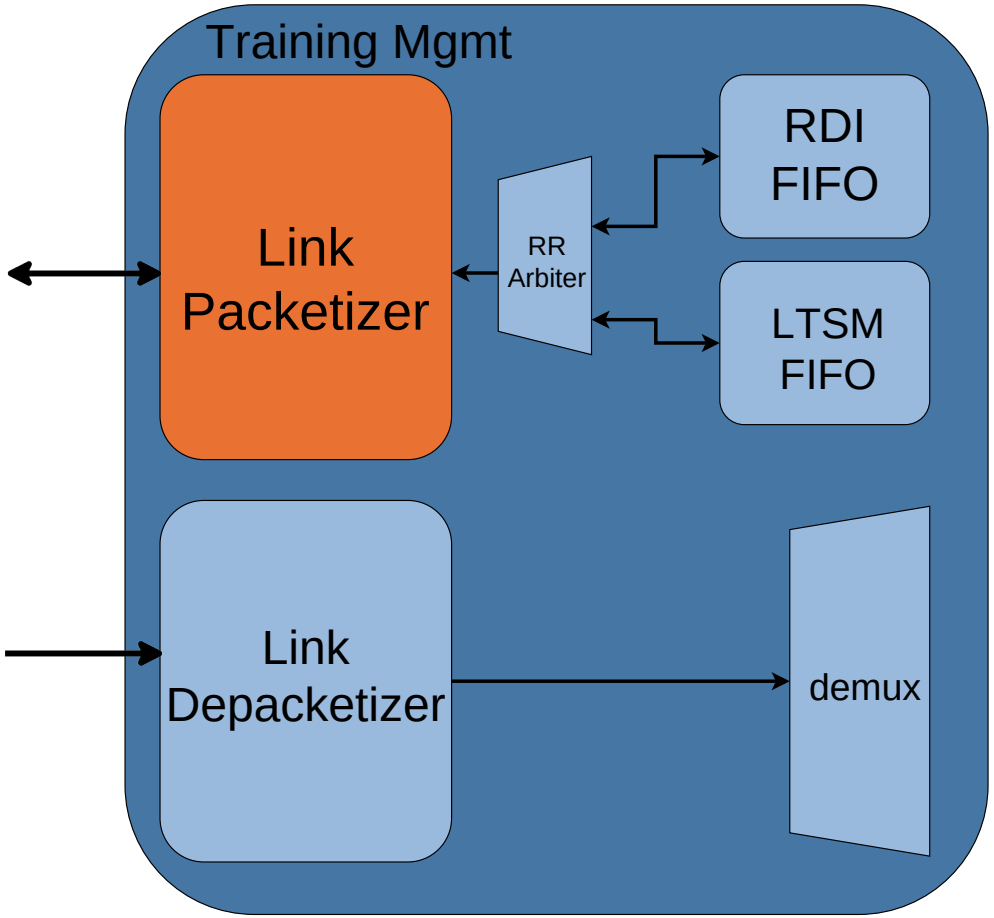
- Fair Arbitration: Round-Robin Arbiter balances LTSM and RDI messages.



SideBand

Training Management

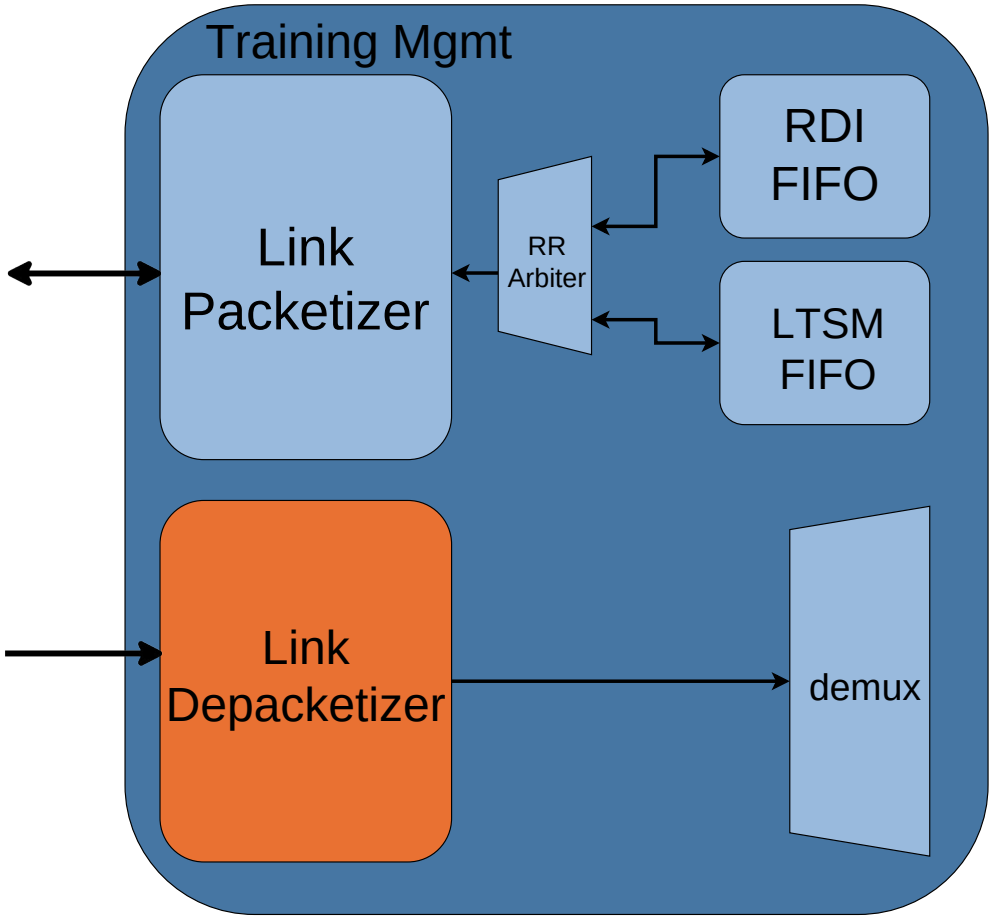
- Packetization & Routing: Assembles and demultiplexes 128-bit control packets.



SideBand

Training Management

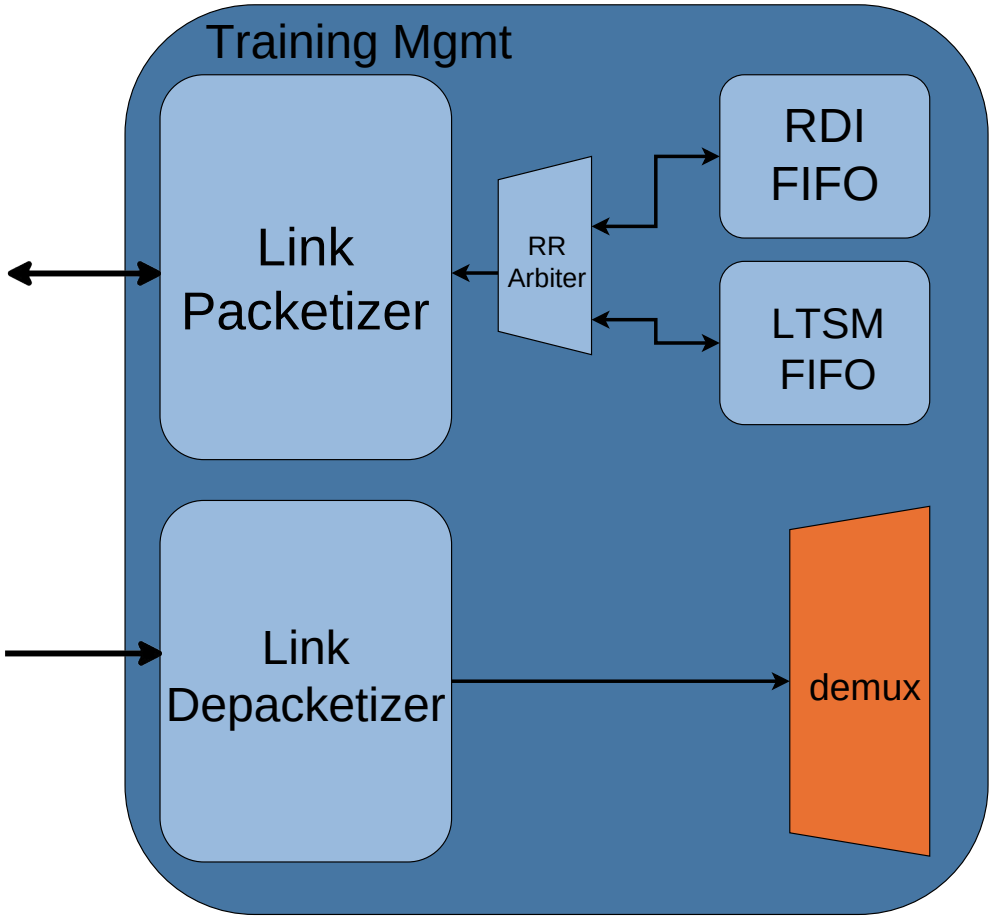
- Link De-packetizer: Parses incoming packets to extract original payloads.



SideBand

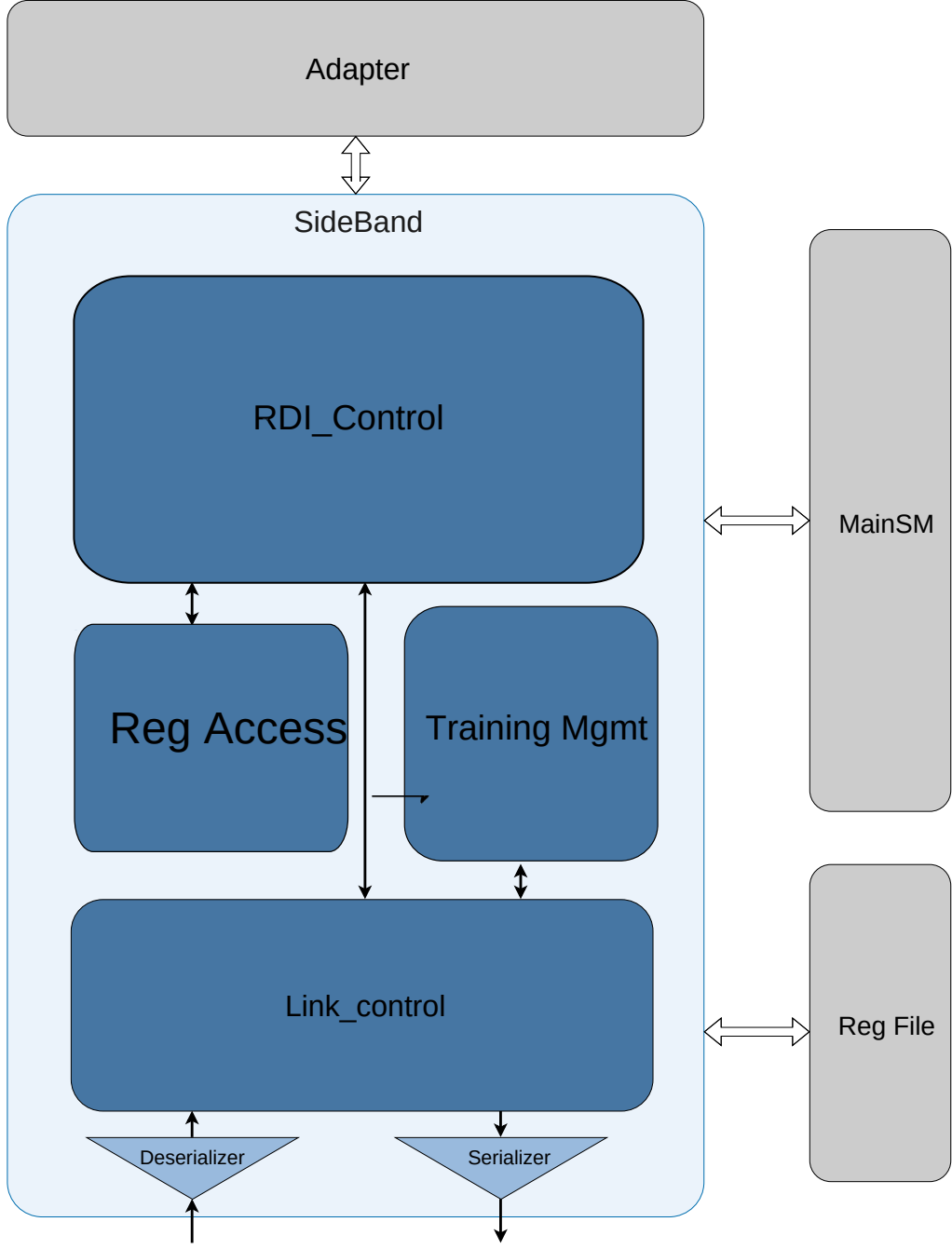
Training Management

- Demux Unit: Routes decoded signals back to the respective Main SM interface.



SideBand

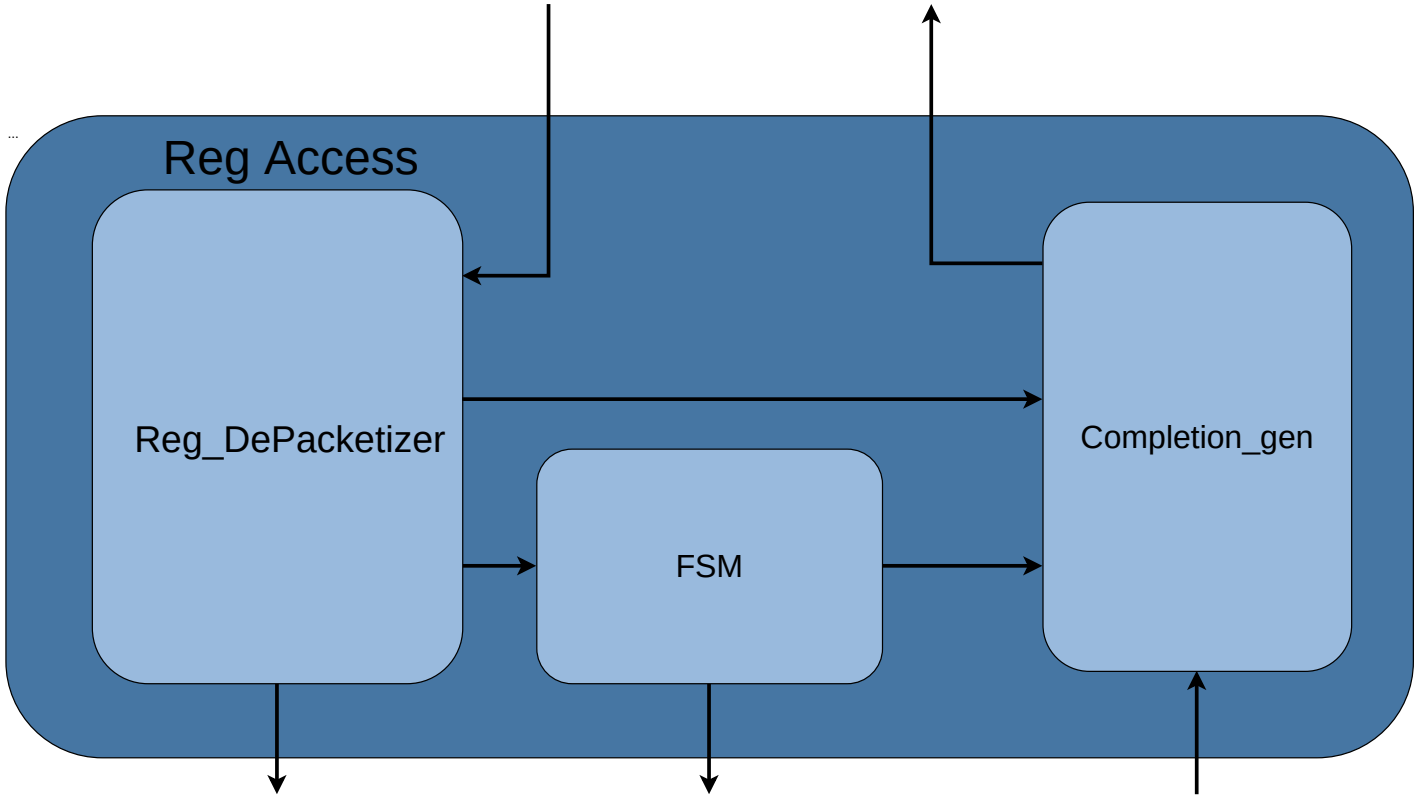
Always-On Control Plane



SideBand

Reg Access

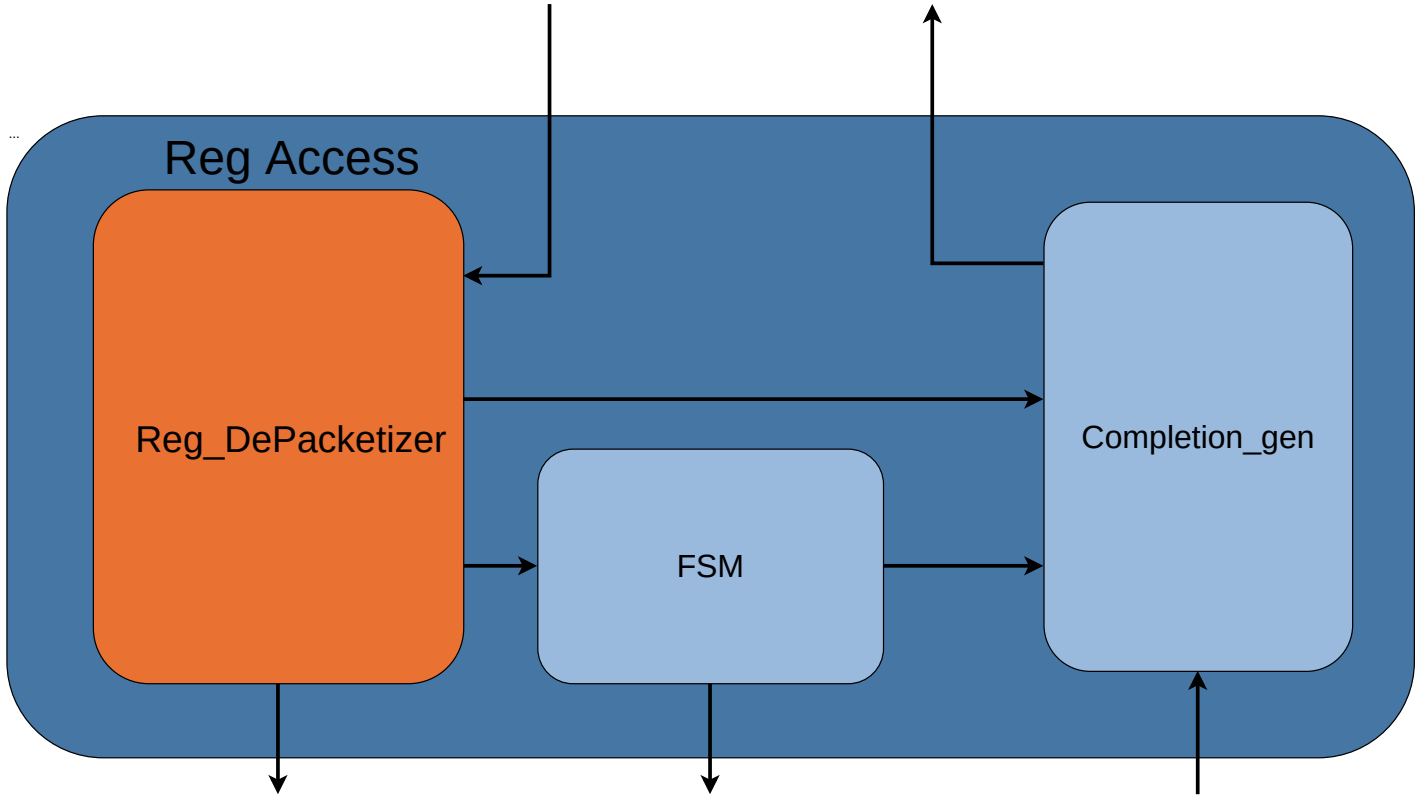
- PHY Register File (RF) Interface: Coordinates read/write access to internal CSRs.



SideBand

Reg Access

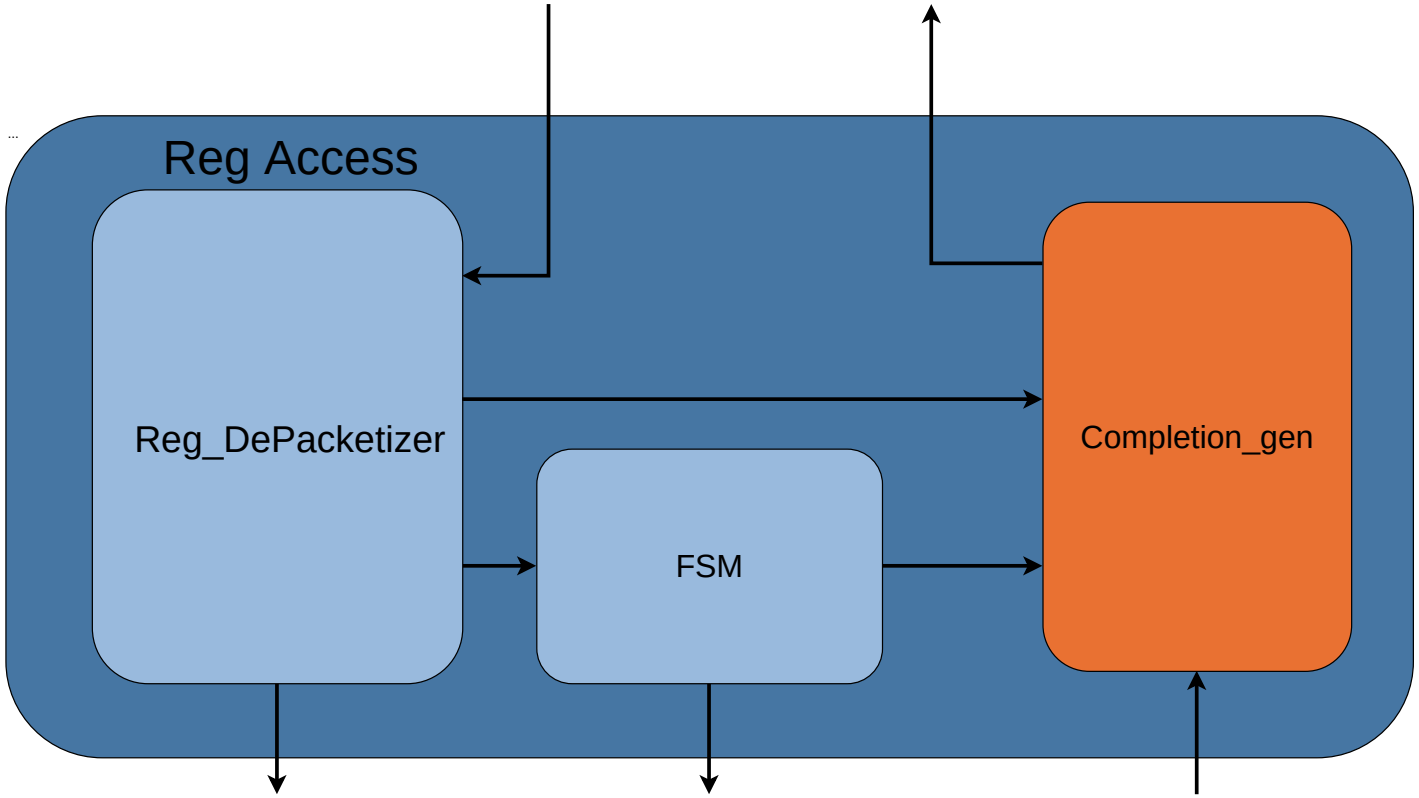
- De-packetizer: Translates 128-bit packets into local RF bus control signals.



SideBand

Reg Access

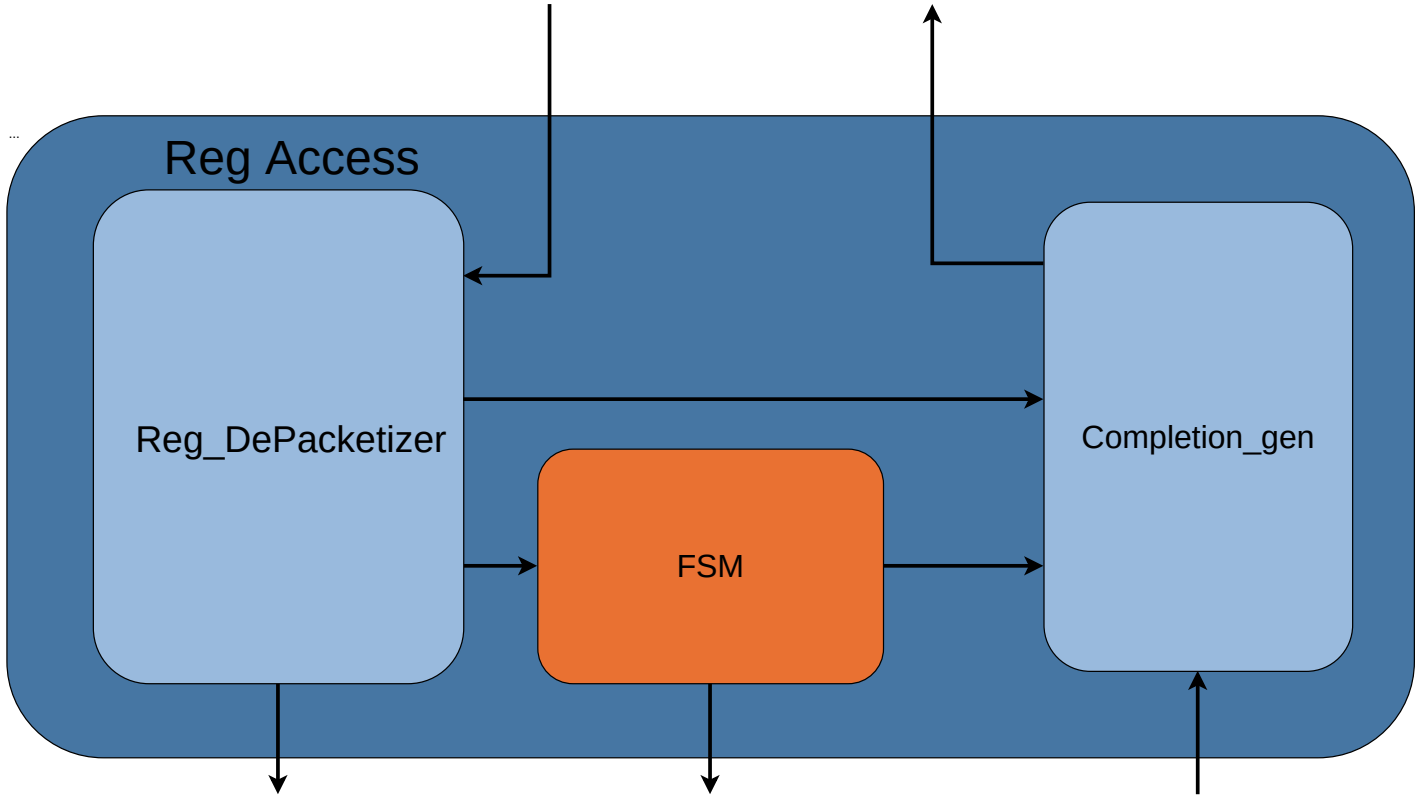
- Completion Generation: Creates response packets with read data and status.



SideBand

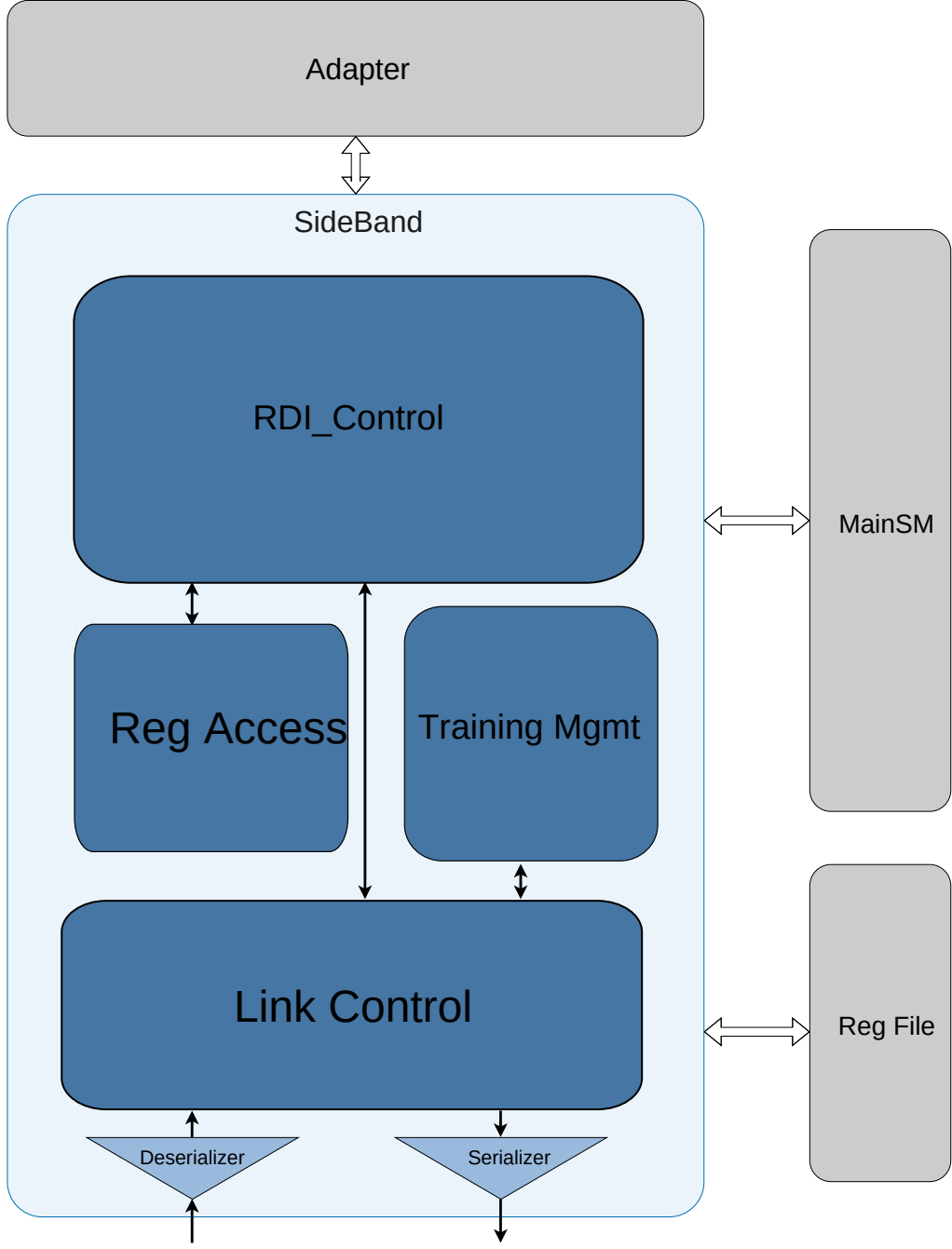
Reg Access

- Control FSM: Sequences the overall decode, execution, and response phases.



SideBand

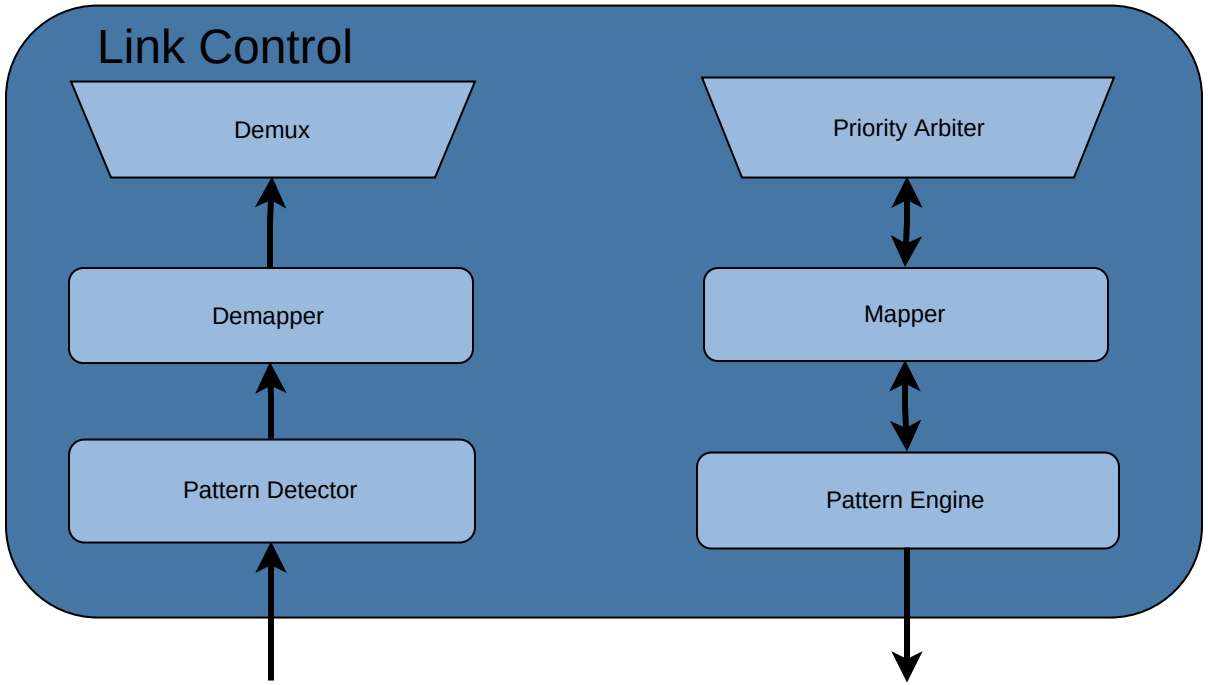
Always-On Control Plane



SideBand

Link Control

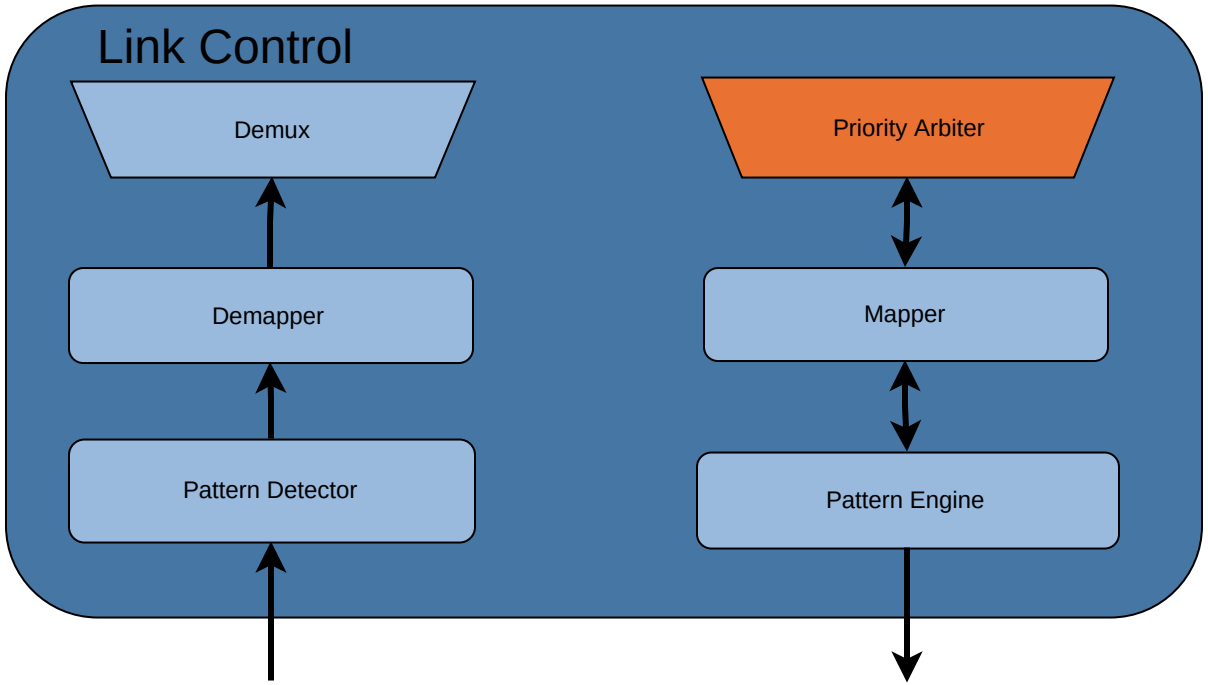
- Physical Link Interface: Coordinates handshakes directly with the SerDes.



SideBand

Link Control

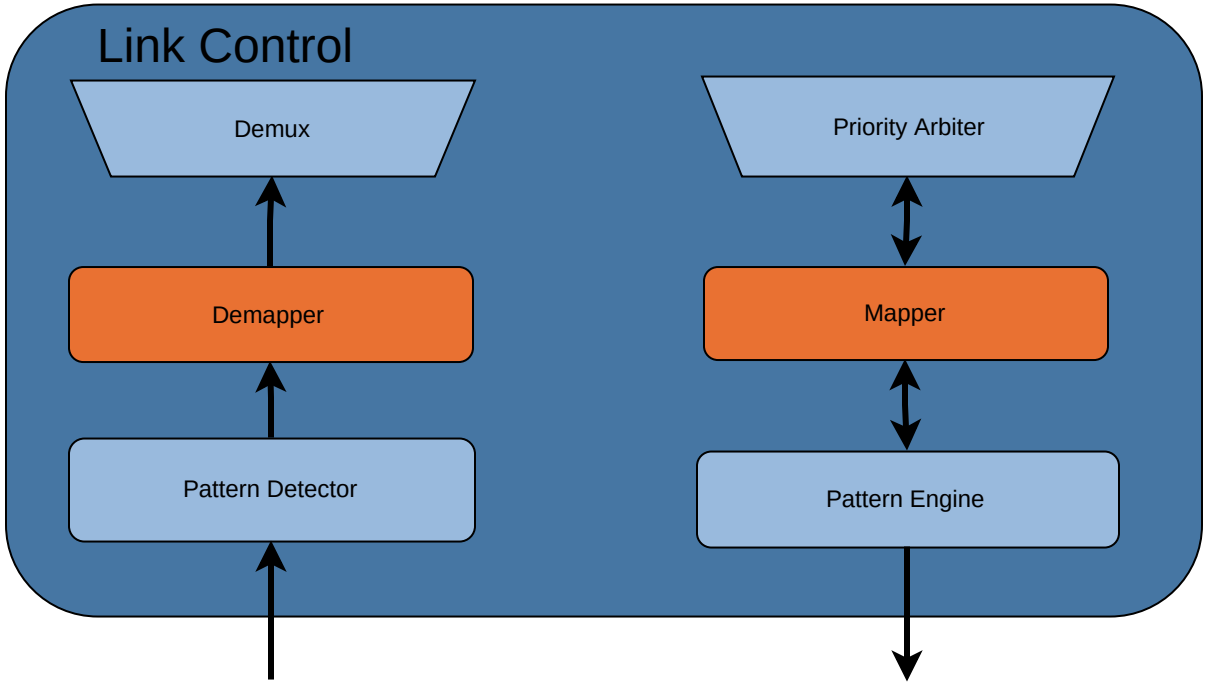
- TX Priority Arbitration: Prioritizes Training/LTSM messages over RDI traffic.



SideBand

Link Control

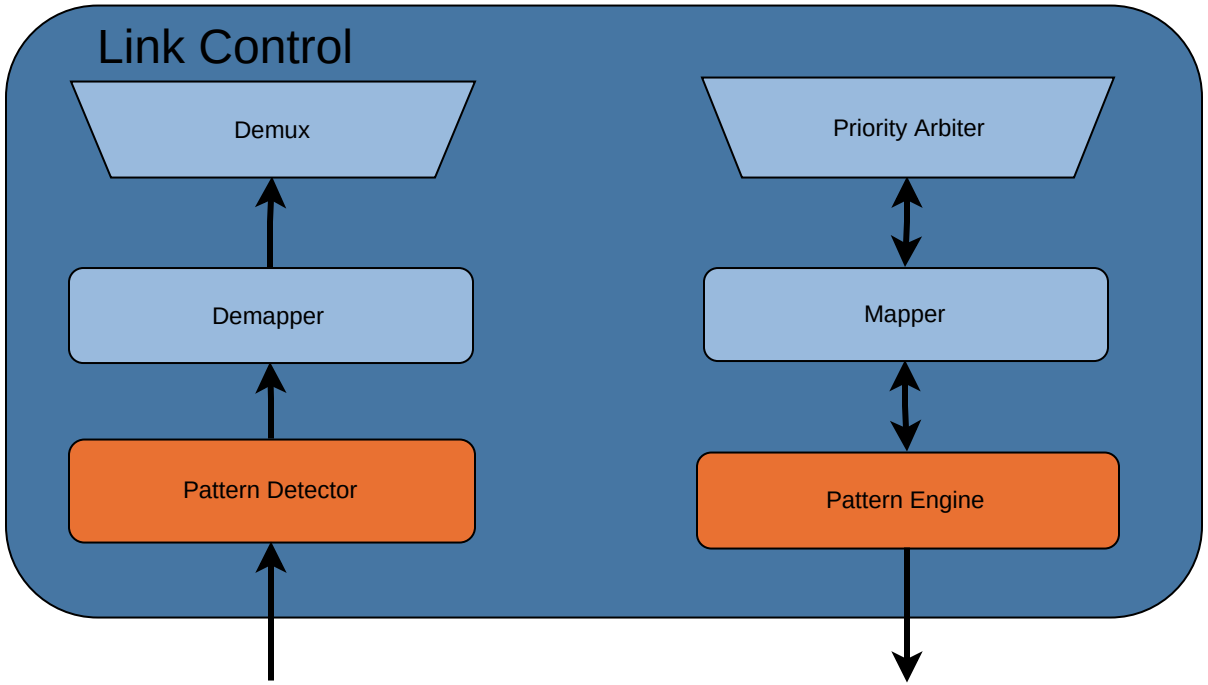
- Width Mapping (128b 🔔 64b): Converts packets for the 64-bit parallel SerDes.



SideBand

Link Control

- SBINIT Handshake: Coordinates pattern engine/detector cycles to lock TX and RX.



SideBand

Always-On Control Plane

Figure 7-1. Format for Register Access Request

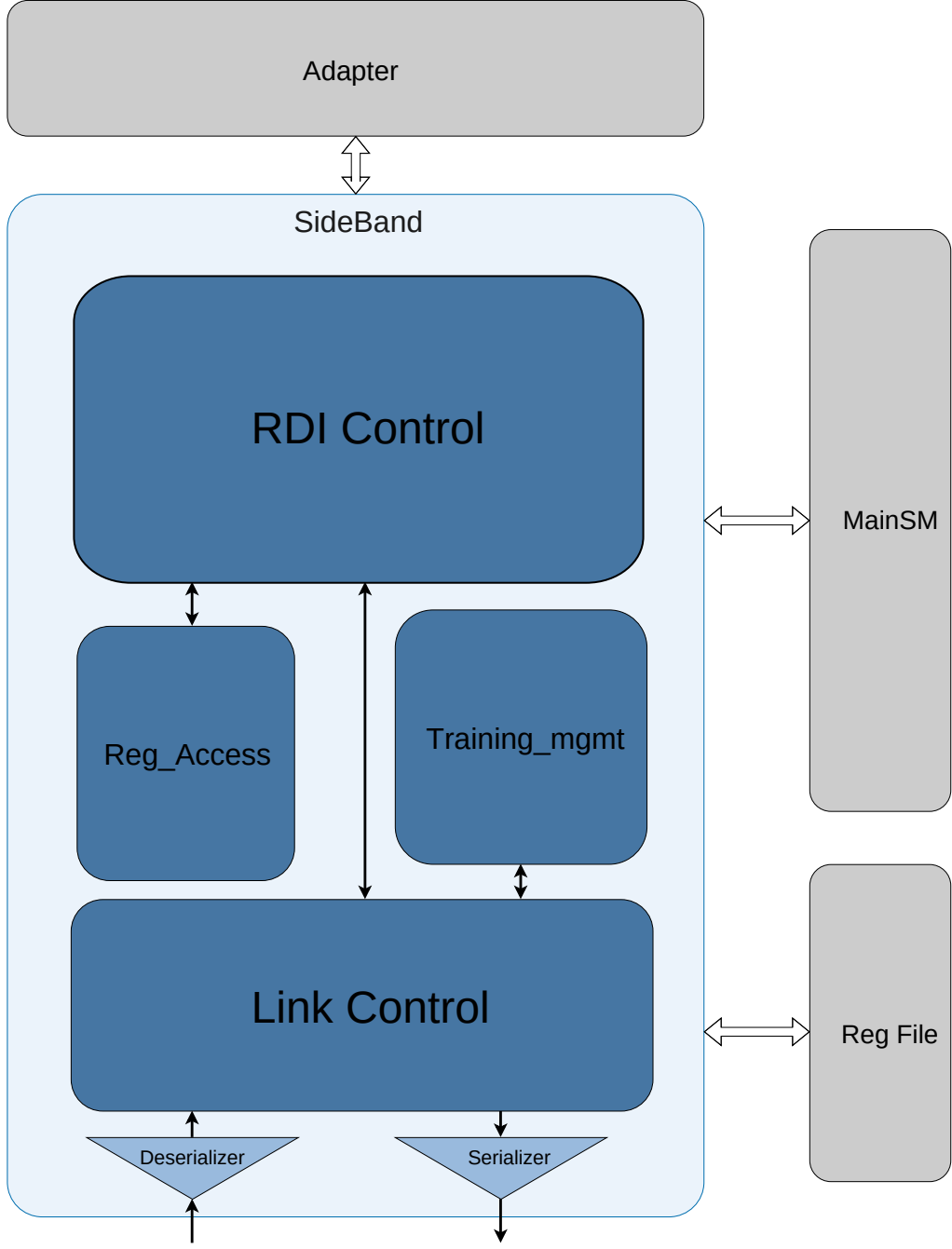
Register Access Request																															
Bytes	3								2								1								0							
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Header / Data	Header																															
Phase0	srcid				rsvd				tag[4:0]				be[7:0]								rsvd							ep	opcode[4:0]			
Phase1	dp	cp	cr		rsvd				dstid				addr[23:0]																			
Header / Data	Data (if applicable, can be 32 bits or 64 bits)																															
Phase2	data[31:0]																															
Phase3	data[63:32]																															

Figure 7-2. Format for Register Access Completions

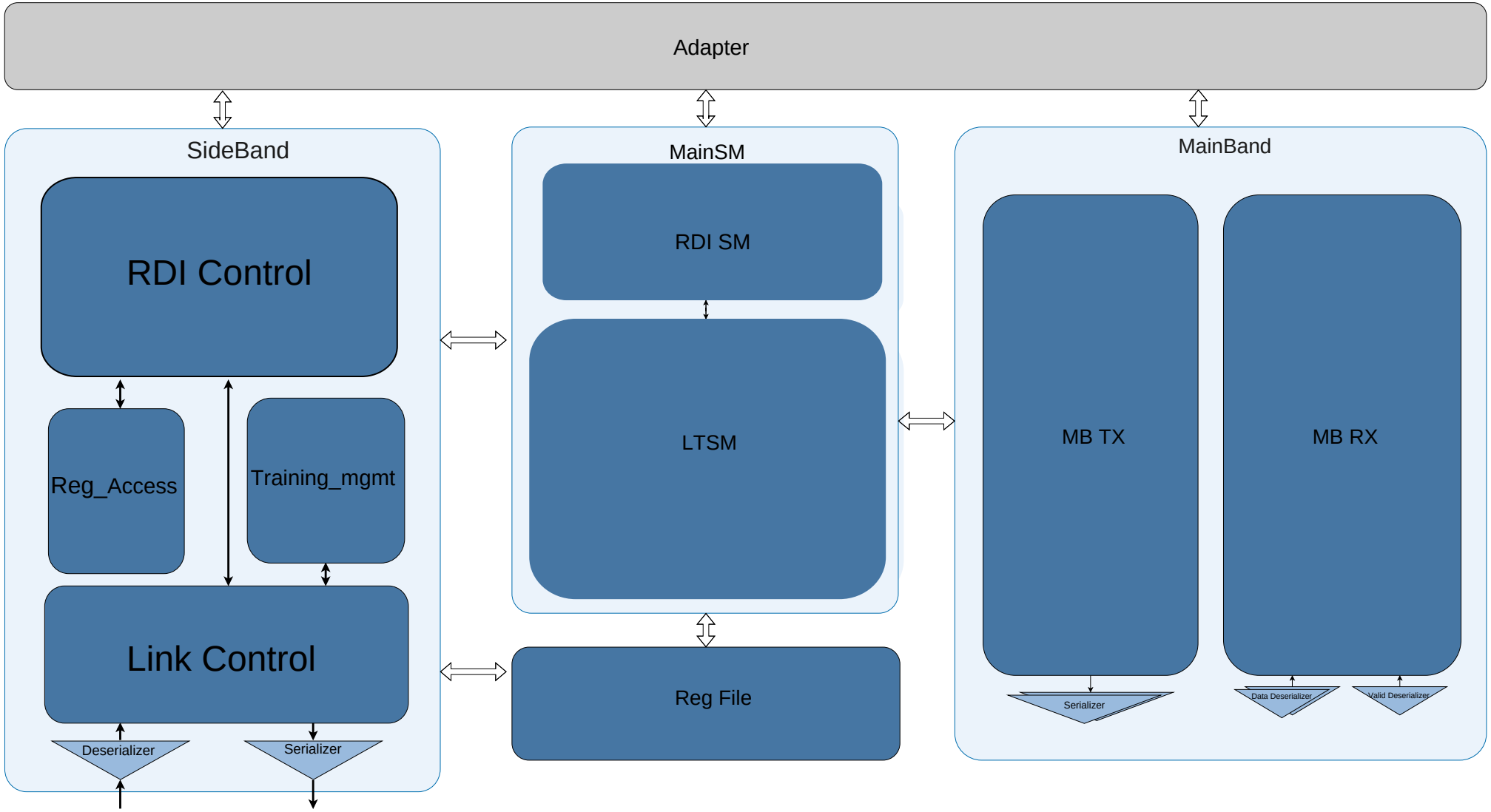
Register Access Completions																																
Bytes	3								2								1								0							
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Header / Data	Header																															
Phase0	srcid				rsvd				tag[4:0]				be[7:0]								rsvd							ep	opcode[4:0]			
Phase1	dp	cp	cr		rsvd				dstid				rsvd																		Status	
Header / Data	Data (if completion with data, can be 32 bits or 64 bits)																															
Phase2	data[31:0]																															
Phase3	data[63:32]																															

Figure 7-4. Format for Messages with data payloads

Messages with data																																
Bytes	3								2								1								0							
Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Header / Data	Header																															
Phase0	srcid				rsvd				rsvd				msgcode[7:0]								rsvd								opcode[4:0]			
Phase1	dp	cp			rsvd				dstid				MsgInfo[15:0]												MsgSubcode[7:0]							
Header / Data	Data																															
Phase2	data[31:0]																															
Phase3	data[63:32]																															



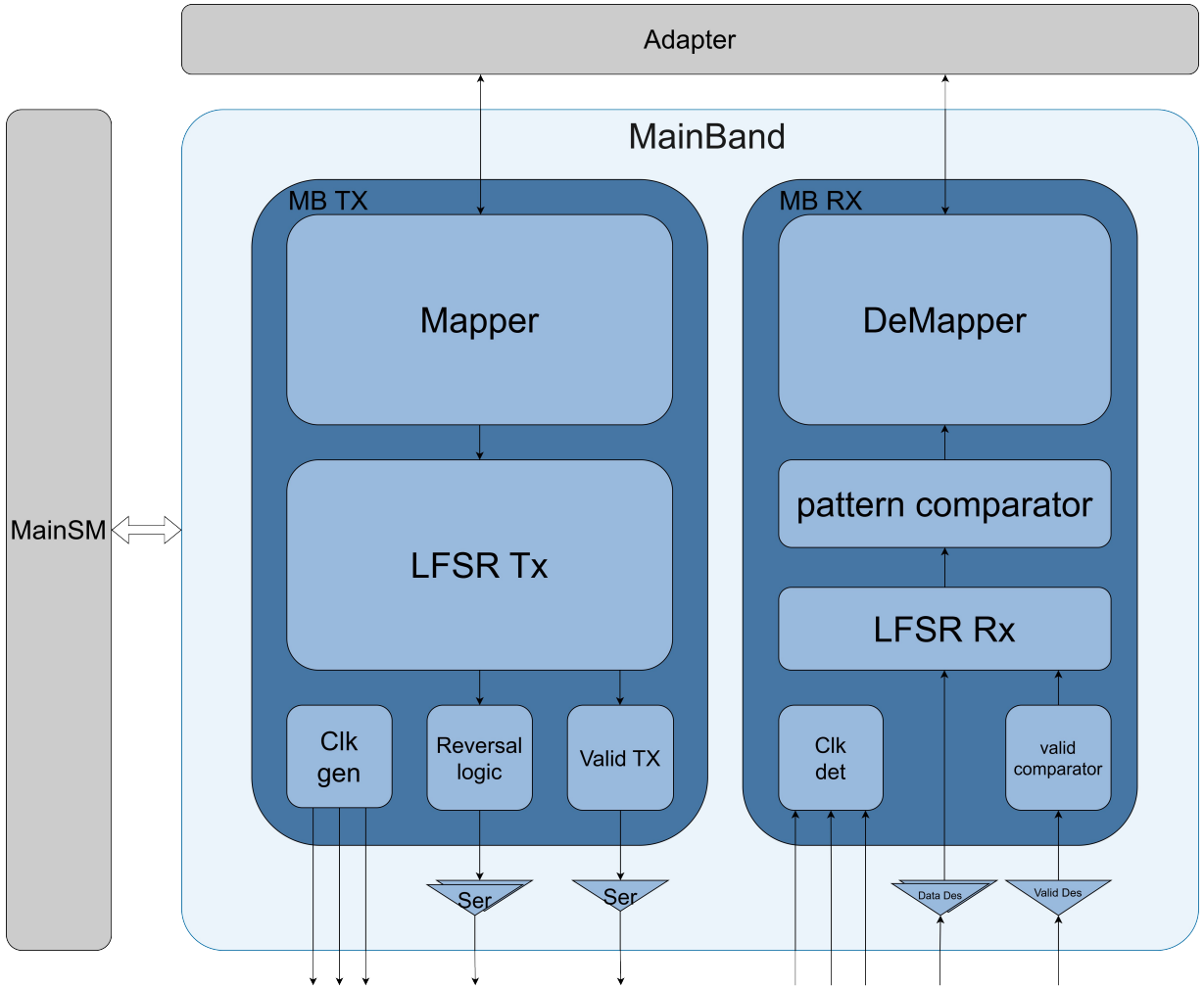
CIe PHY Architecture



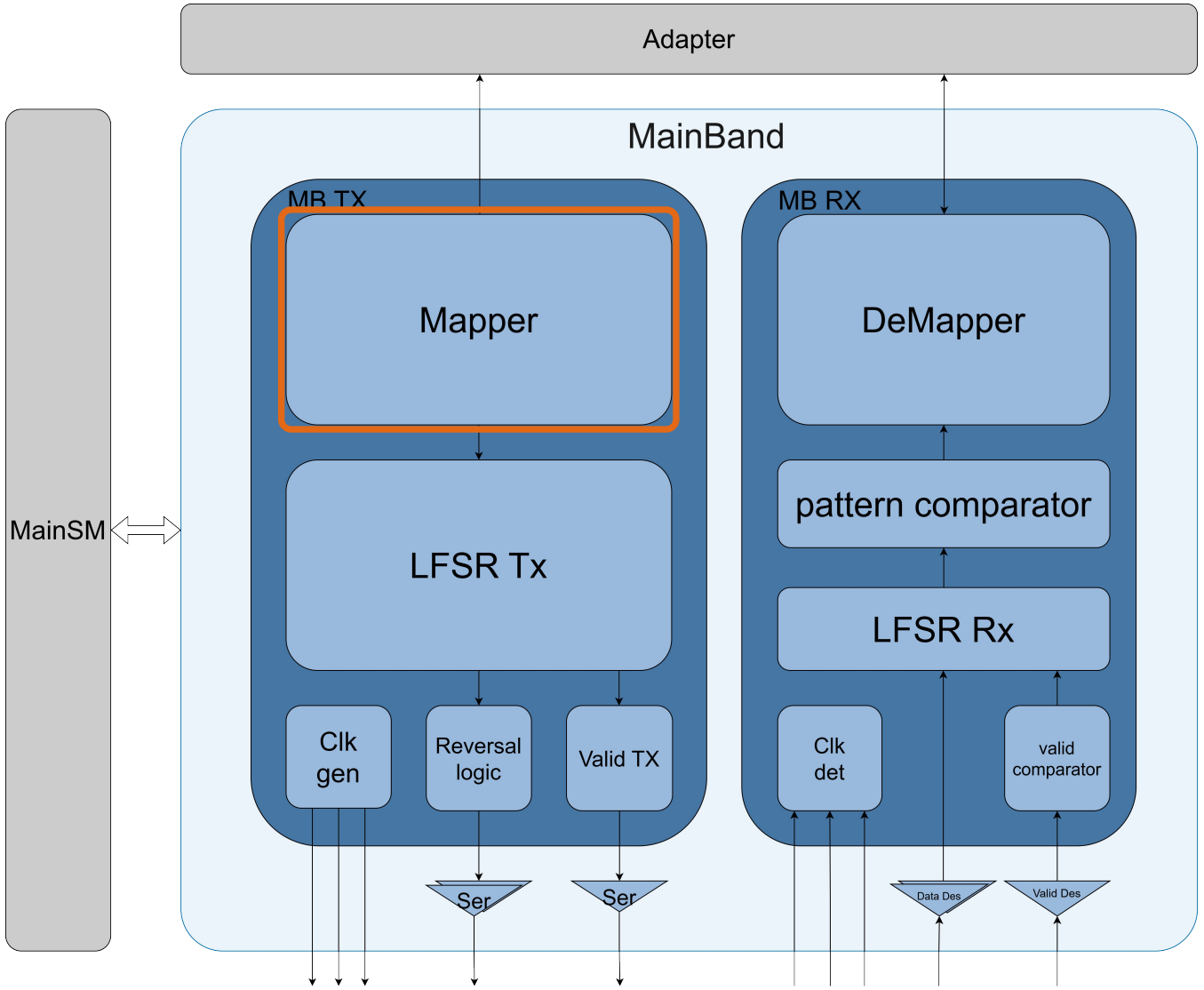
Main Band

Mainband

- Carries the **primary data payload** between dies, mapped and scrambled across active lanes.
- Frames valid data intervals and forwards a clock for receiver sampling.
- Supports **width degradation**, **lane reversal**, and **link training** under MainSM control.



Mainband

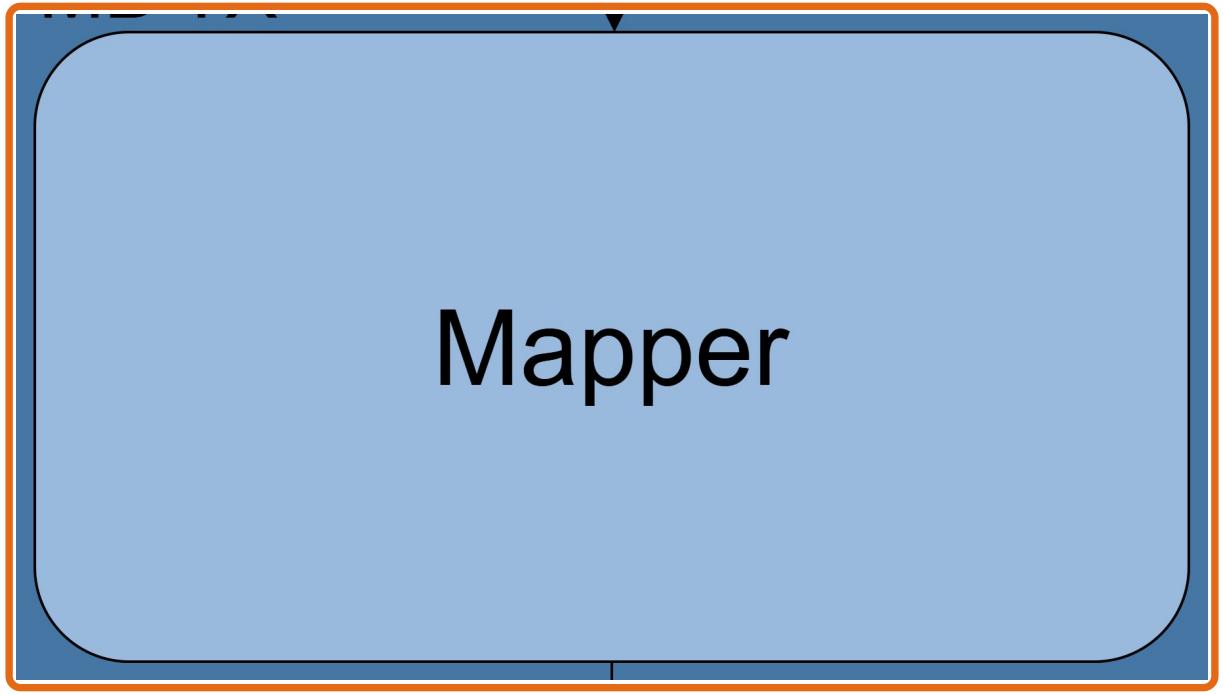

Adapter
MainBand
MB TX
Mapper
LFSR Tx
Clk gen
Reversal logic
Valid TX
Ser
Ser
MB RX
DeMapper
pattern comparator
LFSR Rx
Clk det
valid comparator
Data Des
Valid Des
MainSM

Mapper

•Entry point of the **TX data path**, receiving the512-bit adapter word each clock cycle.

- **Splits the data evenly into 16 parallel 32-bit lane streams.**

- Supports **width-degraded modes(x16/x8/x4**) and forces outputs to zero when disabled.



Mapper

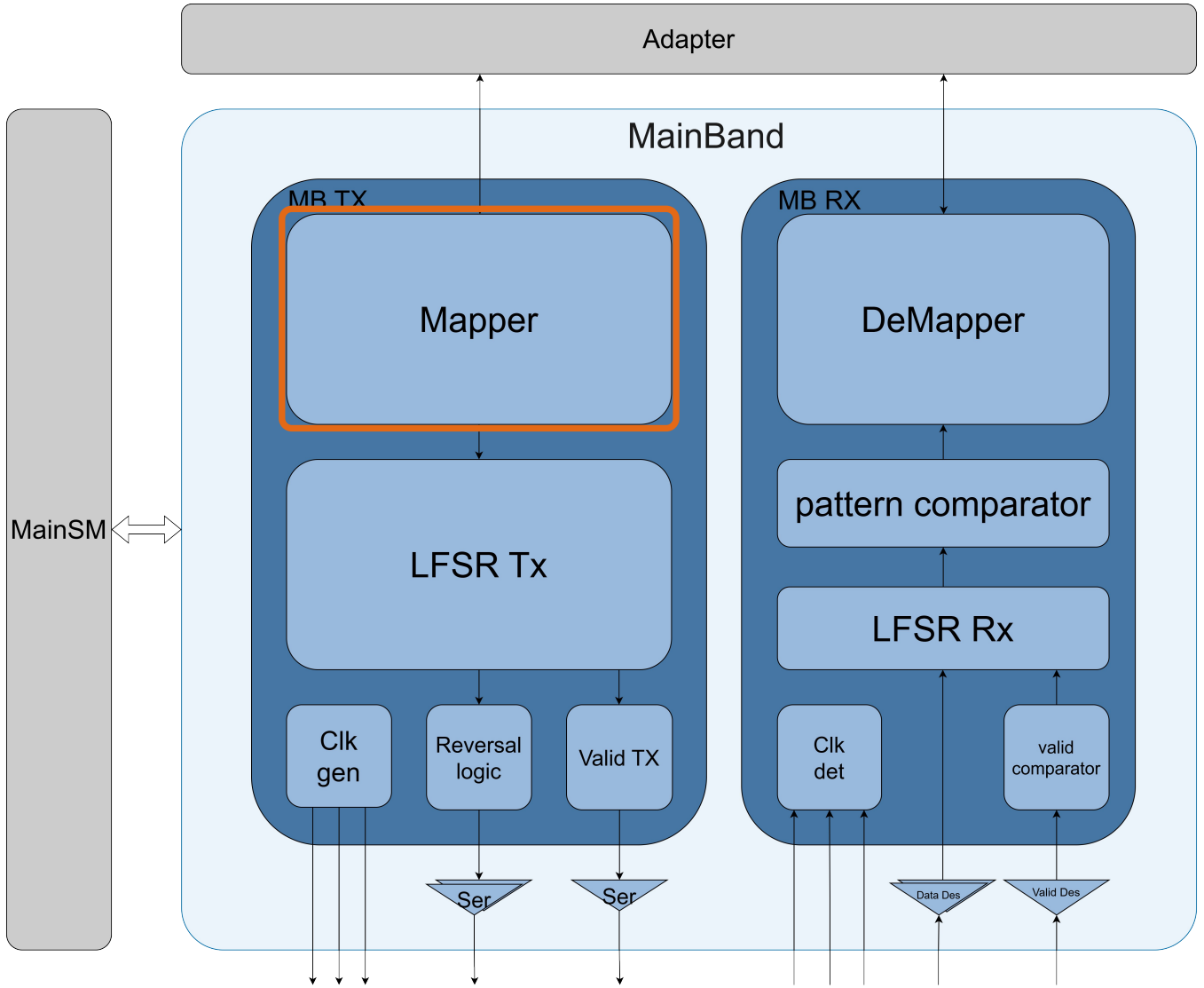
UI \ Lane	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 - 7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
8 - 15	B16	B17	B18	B19	B20	B21	B22	B23	B24	B25	B26	B27	B28	B29	B30	B31
16 - 23	B32	B33	B34	B35	B36	B37	B38	B39	B40	B41	B42	B43	B44	B45	B46	B47
24 - 31	B48	B49	B50	B51	B52	B53	B54	B55	B56	B57	B58	B59	B60	B61	B62	B63
32 - 39	B64	B65	B66	B67	B68	B69	B70	B71	B72	B73	B74	B75	B76	B77	B78	B79
40 - 47	B80	B81	B82	B83	B84	B85	B86	B87	B88	B89	B90	B91	B92	B93	B94	B95
48 - 55	B96	B97	B98	B99	B100	B101	B102	B103	B104	B105	B106	B107	B108	B109	B110	B111
56 - 63	B112	B113	B114	B115	B116	B117	B118	B119	B120	B121	B122	B123	B124	B125	B126	B127
64 - 71	B128	B129	B130	B131	B132	B133	B134	B135	B136	B137	B138	B139	B140	B141	B142	B143
72 - 79	B144	B145	B146	B147	B148	B149	B150	B151	B152	B153	B154	B155	B156	B157	B158	B159
80 - 87	B160	B161	B162	B163	B164	B165	B166	B167	B168	B169	B170	B171	B172	B173	B174	B175
88 - 95	B176	B177	B178	B179	B180	B181	B182	B183	B184	B185	B186	B187	B188	B189	B190	B191
96 - 103	B192	B193	B194	B195	B196	B197	B198	B199	B200	B201	B202	B203	B204	B205	B206	B207
104 - 111	B208	B209	B210	B211	B212	B213	B214	B215	B216	B217	B218	B219	B220	B221	B222	B223
112 - 119	B224	B225	B226	B227	B228	B229	B230	B231	B232	B233	B234	B235	B236	B237	B238	B239
120 - 127	B240	B241	B242	B243	B244	B245	B246	B247	B248	B249	B250	B251	B252	B253	B254	B255

Lane	8	9	10	11	12	13	14	15
	or							
UI \ Lane	0	1	2	3	4	5	6	7
0 - 7	B0	B1	B2	B3	B4	B5	B6	B7
8 - 15	B8	B9	B10	B11	B12	B13	B14	B15
16 - 23	B16	B17	B18	B19	B20	B21	B22	B23
24 - 31	B24	B25	B26	B27	B28	B29	B30	B31
32 - 39	B32	B33	B34	B35	B36	B37	B38	B39
40 - 47	B40	B41	B42	B43	B44	B45	B46	B47
48 - 55	B48	B49	B50	B51	B52	B53	B54	B55
56 - 63	B56	B57	B58	B59	B60	B61	B62	B63

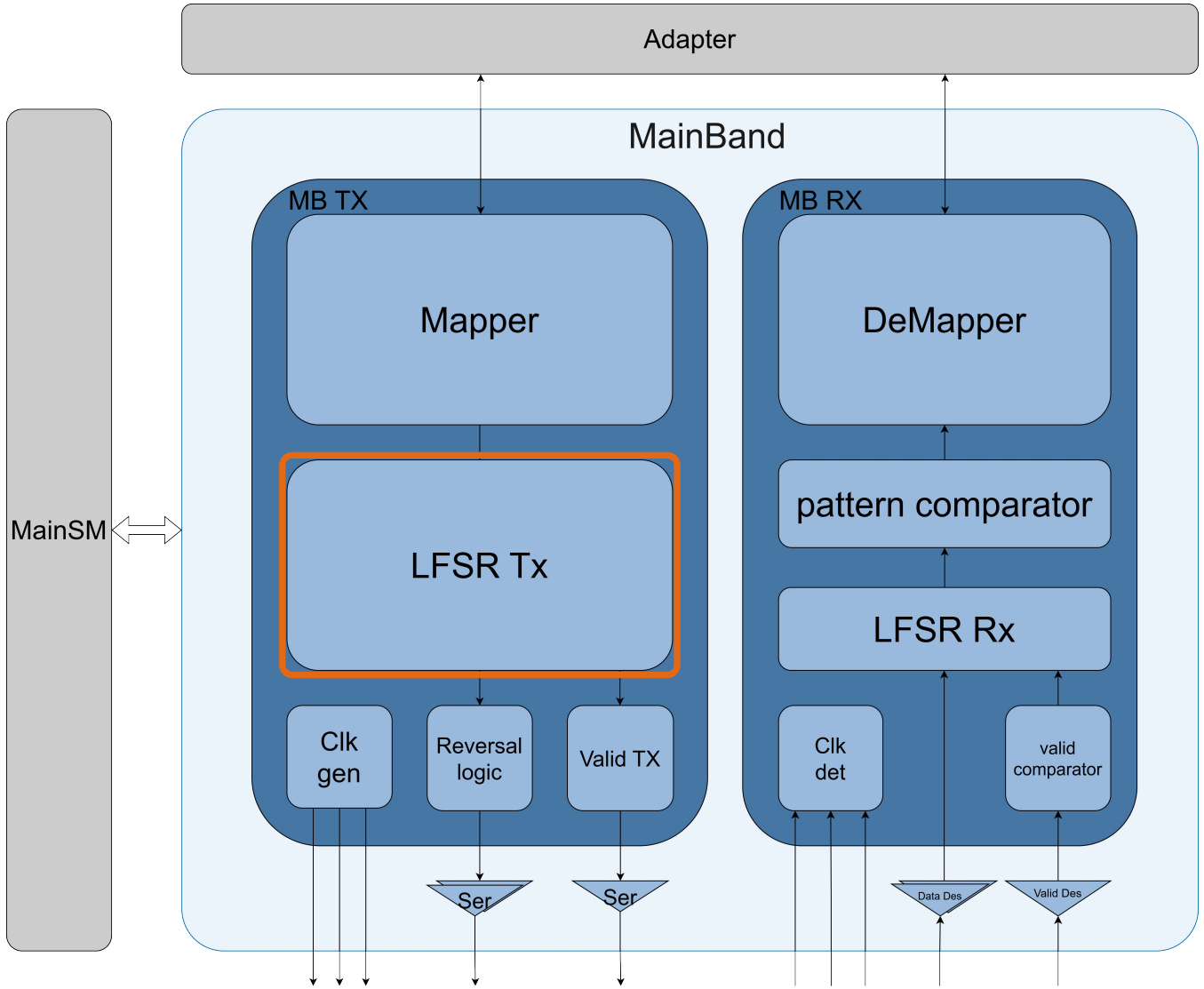
Mapper

UI \ Lane	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 - 7	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
8 - 15	B16	B17	B18	B19	B20	B21	B22	B23	B24	B25	B26	B27	B28	B29	B30	B31
16 - 23	B32	B33	B34	B35	B36	B37	B38	B39	B40	B41	B42	B43	B44	B45	B46	B47
24 - 31	B48	B49	B50	B51	B52	B53	B54	B55	B56	B57	B58	B59	B60	B61	B62	B63

Mainband

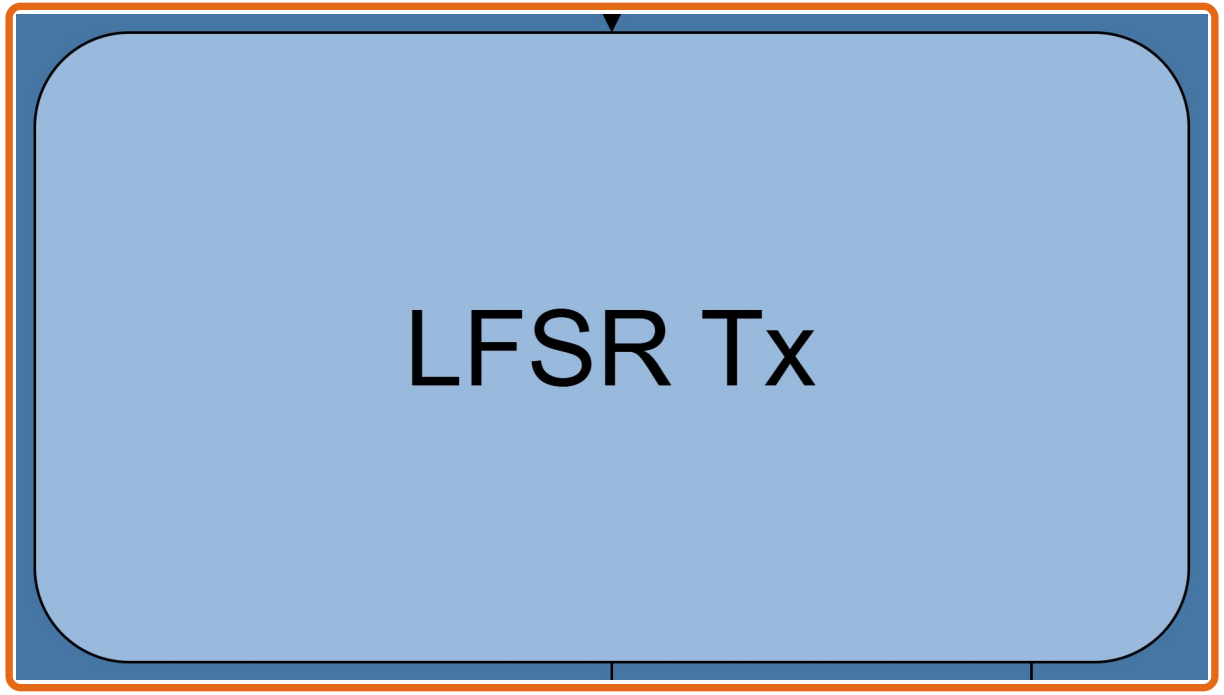

Adapter
MainSM
MainBand
MB TX
Mapper
LFSR Tx
Clk gen
Reversal logic
Valid TX
Ser
Ser
MB RX
DeMapper
pattern comparator
LFSR Rx
Clk det
valid comparator
Data Des
Valid Des

Mainband


Adapter
MainBand
MainSM
MB TX
Mapper
LFSR Tx
Clk gen
Reversal logic
Valid TX
Ser
Ser
MB RX
DeMapper
pattern comparator
LFSR Rx
Clk det
valid comparator
Data Des
Valid Des

LFSR TX

- **Scrambles payload data via XOR** with an **LFSR sequence** during **active operation** to reduce DC wander.
- Reuses the same hardware to generate **PRBS-23** and **Per-Lane ID** test patterns during **link training**.
- **Seeds LFSRs** deterministically at each training state to keep TX/RX phase-aligned.



LFSR TX

- The PRBS used is based on a 23-stage maximal-length LFSR with the polynomial:

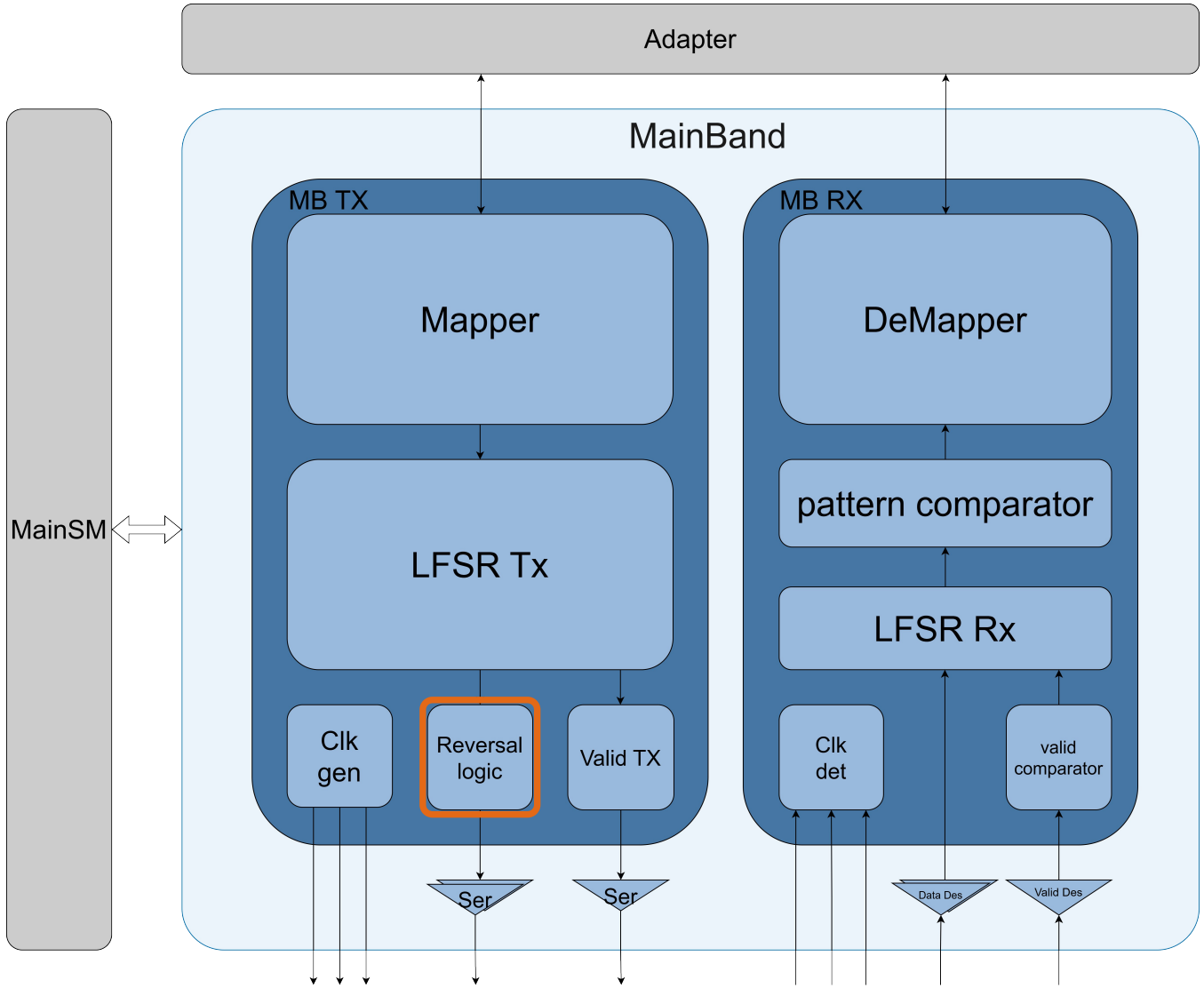
$G(x) = x^{23} + x^{21} + x^{16} + x^{8} + x^{5} + x^{2} + 1$

Lane	Seed Value (Hex)
0	23'h1DBFBC
1	23'h0607BB
2	23'h1EC760
3	23'h18C0DB
4	23'h010F12
5	23'h19CFC9
6	23'h0277CE
7	23'h1BB807

Lane ID: Standard Package Module

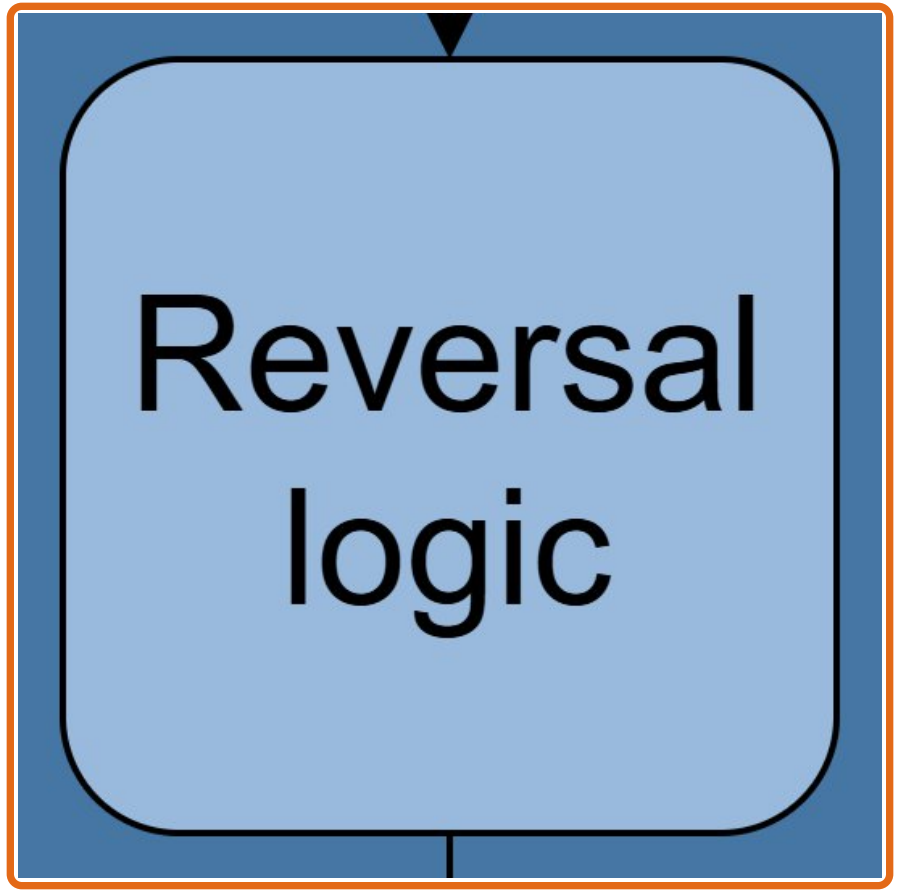
Logical Lane Number	Lane ID	Logical Lane Number	Lane ID
0	00000000b	8	00001000b
1	00000001b	9	00001001b
2	00000010b	10	00001010b
3	00000011b	11	00001011b
4	00000100b	12	00001100b
5	00000101b	13	00001101b
6	00000110b	14	00001110b
7	00000111b	15	00001111b

Mainband

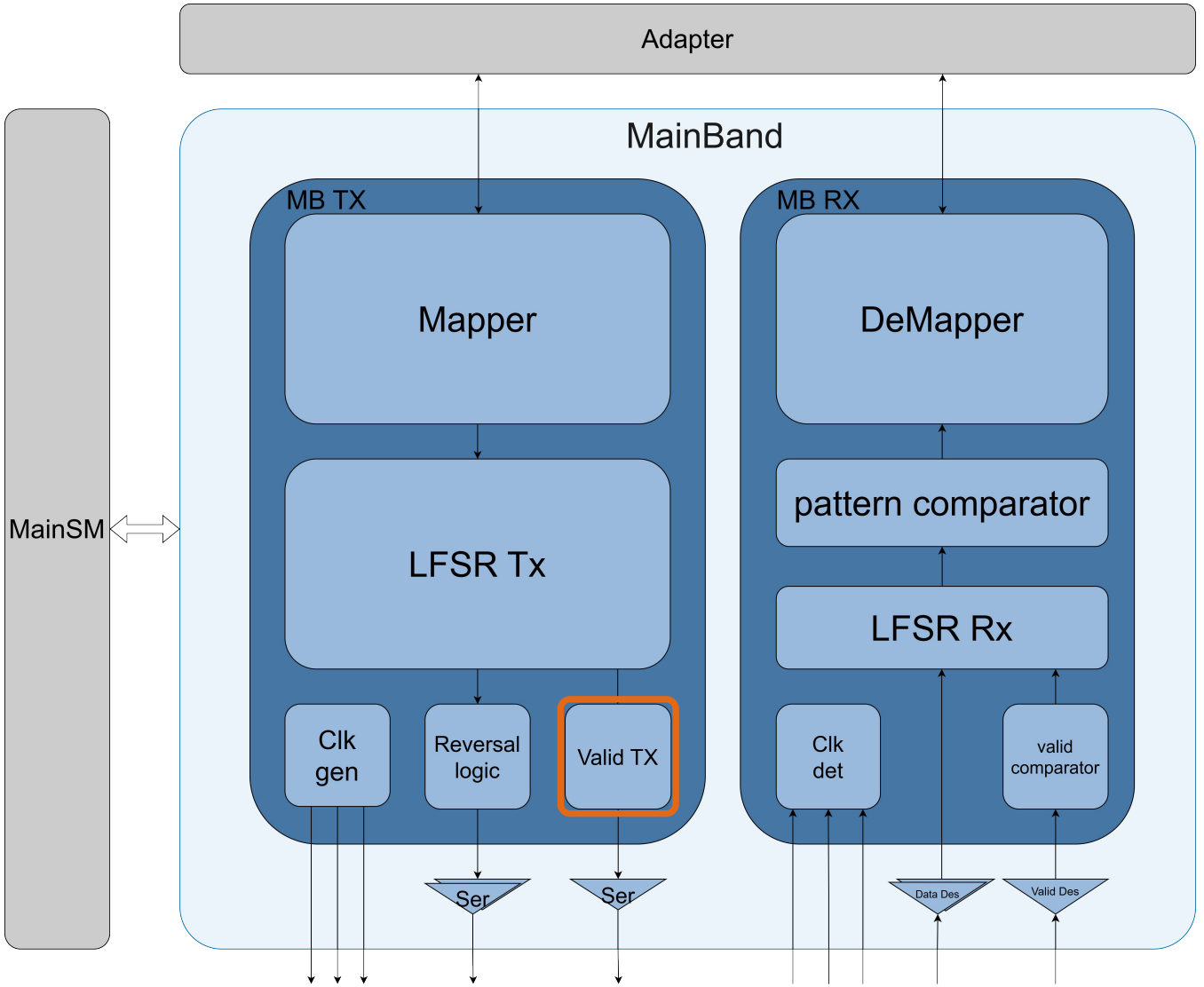

Adapter
MainSM
MainBand
MB TX
Mapper
LFSR Tx
Clk gen
Reversal logic
Valid TX
Ser
Ser
MB RX
DeMapper
pattern comparator
LFSR Rx
Clk det
valid comparator
Data Des
Valid Des

Reversal Logic

- **Re-orders the TX lane** assignment when lane reversal is detected during (**MBINIT.REVERSALMB)**
- Applies the **reversal transparently** so the **remote RX** samples lanes in logical order.
- Acts as a **pass**-through when the link is not reversed.

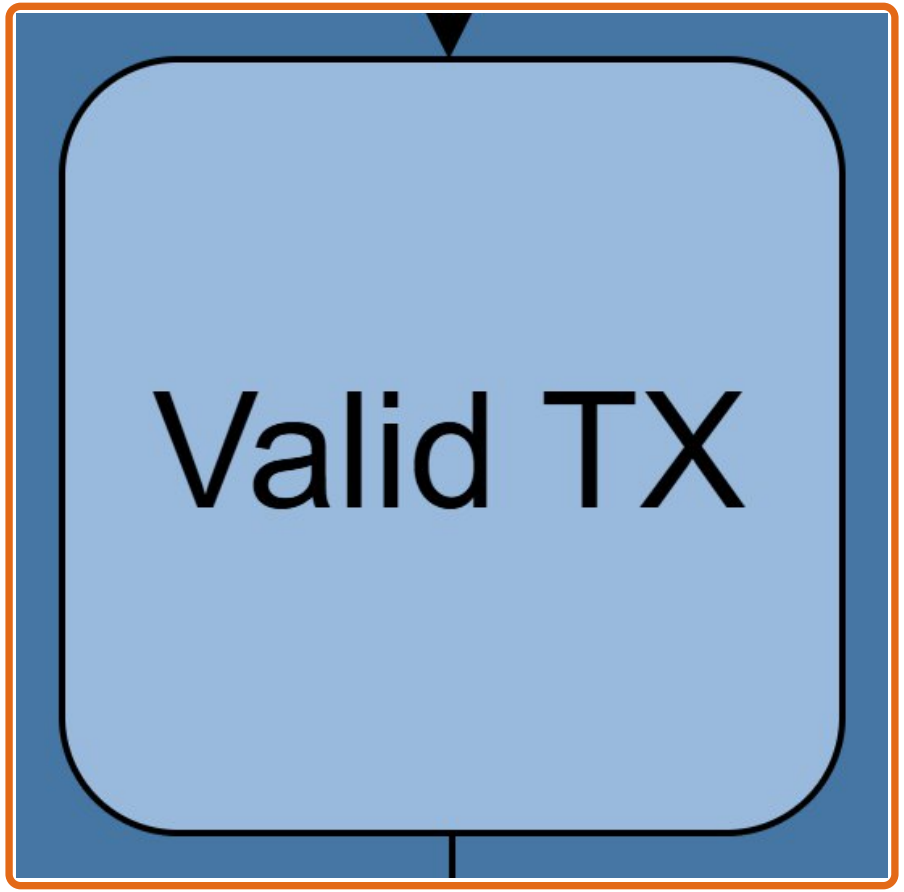


Mainband

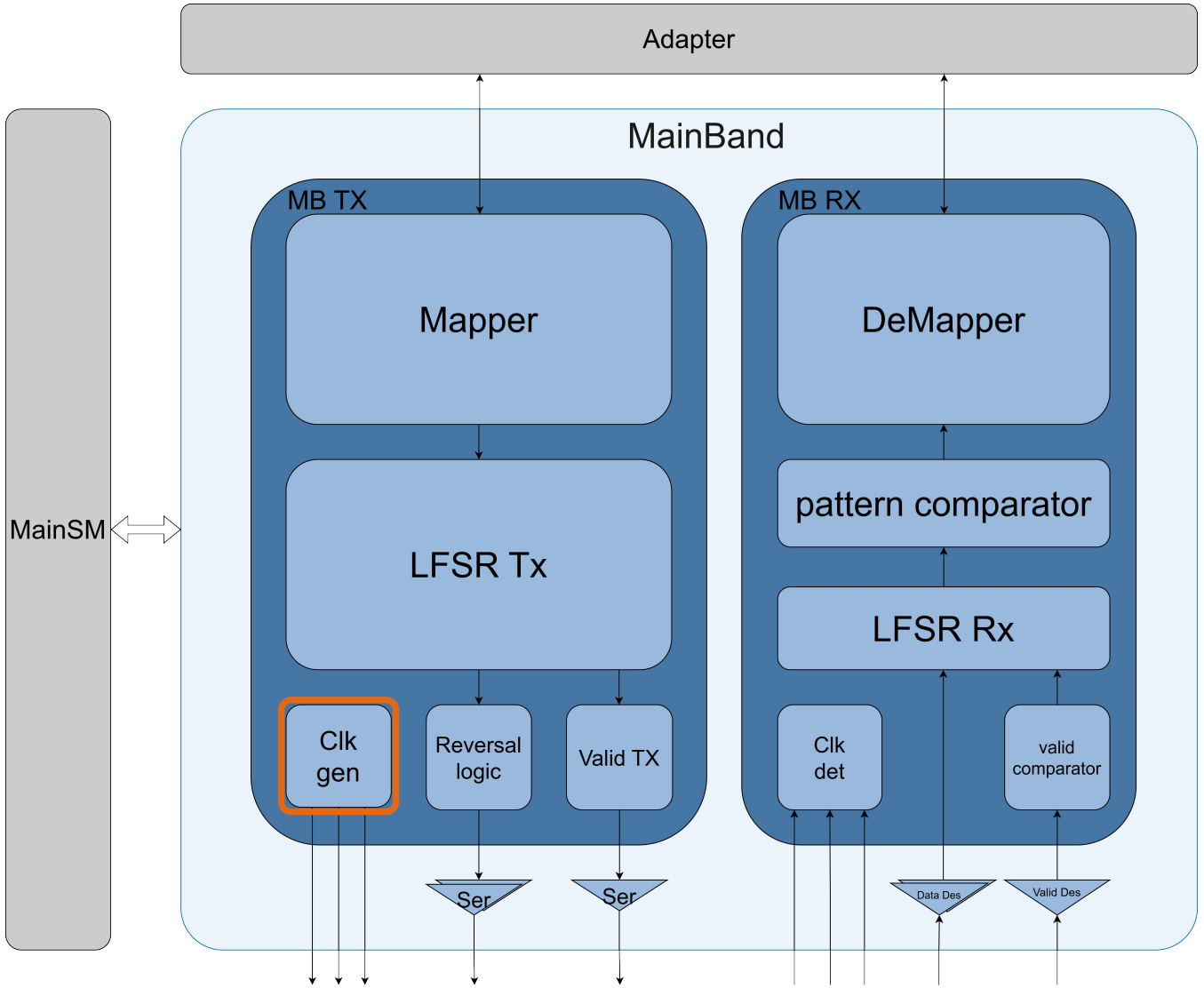

Adapter
MainSM
MainBand
MB TX
Mapper
LFSR Tx
Clk gen
Reversal logic
Valid TX
Ser
Ser
MB RX
DeMapper
pattern comparator
LFSR Rx
Clk det
valid comparator
Data Des
Valid Des

Valid TX

- Drives the valid lane with the fixed framing pattern **(4 UI high / 4 UI low).**
- **Encodes** the **valid framing pattern** used by the **receiver** to qualify incoming data.
- Sequenced by MainSM across training states and mission mode.

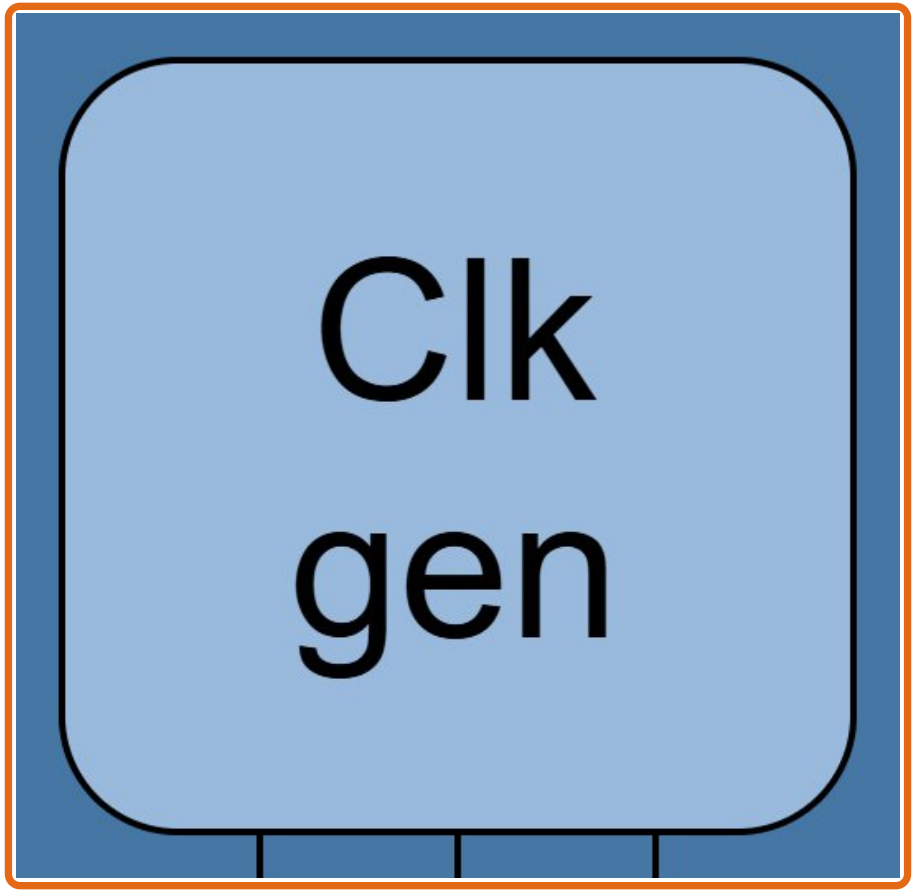


Mainband

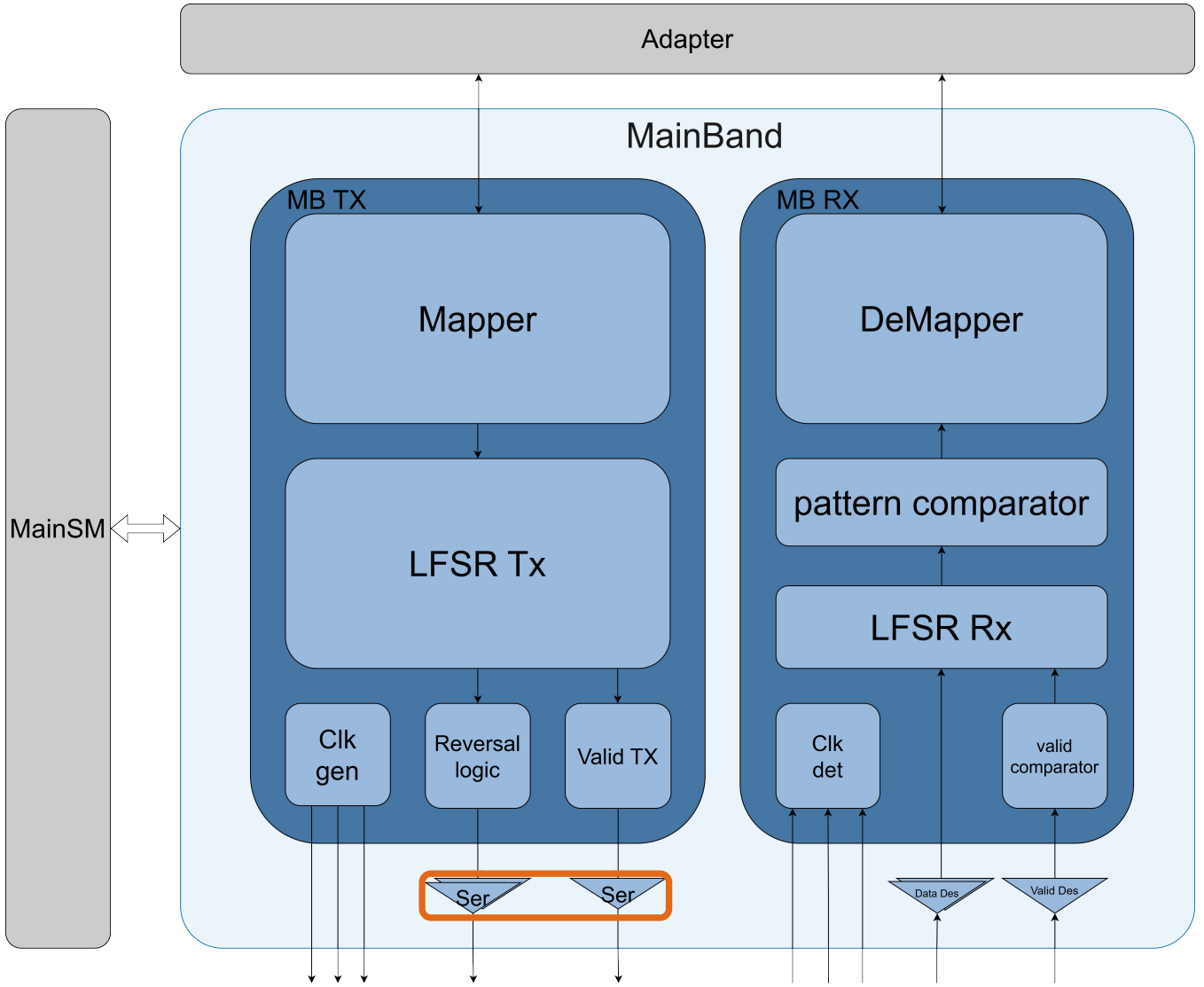

Adapter
MainBand
MainSM
MB TX
Mapper
LFSR Tx
Clk gen
Reversal logic
Valid TX
Ser
Ser
MB RX
DeMapper
pattern comparator
LFSR Rx
Clk det
valid comparator
Data Des
Valid Des

Clock Pattern Generation

- Drives the **forwarded differential** clock **(TCKP/TCKN)** and track lane **(TTRK)** signals.
- Transmits a **128-iteration clock** repair pattern during (**MBINIT.REPAIRCLK)** training.
- Continuously forwards the **PLL-derived** clock to the remote receiver in normal operation.

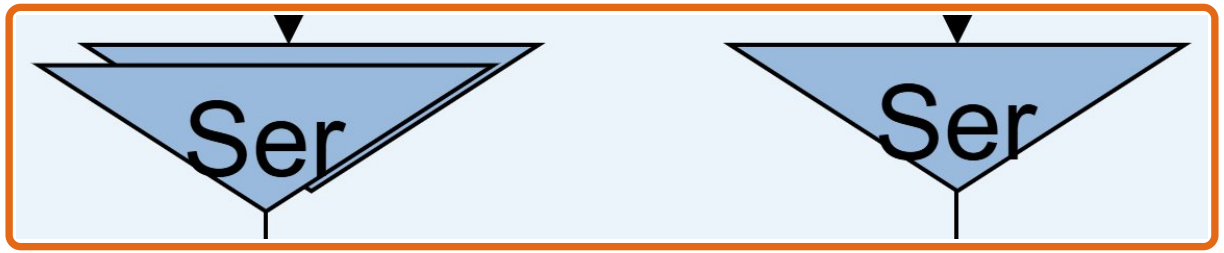


Mainband

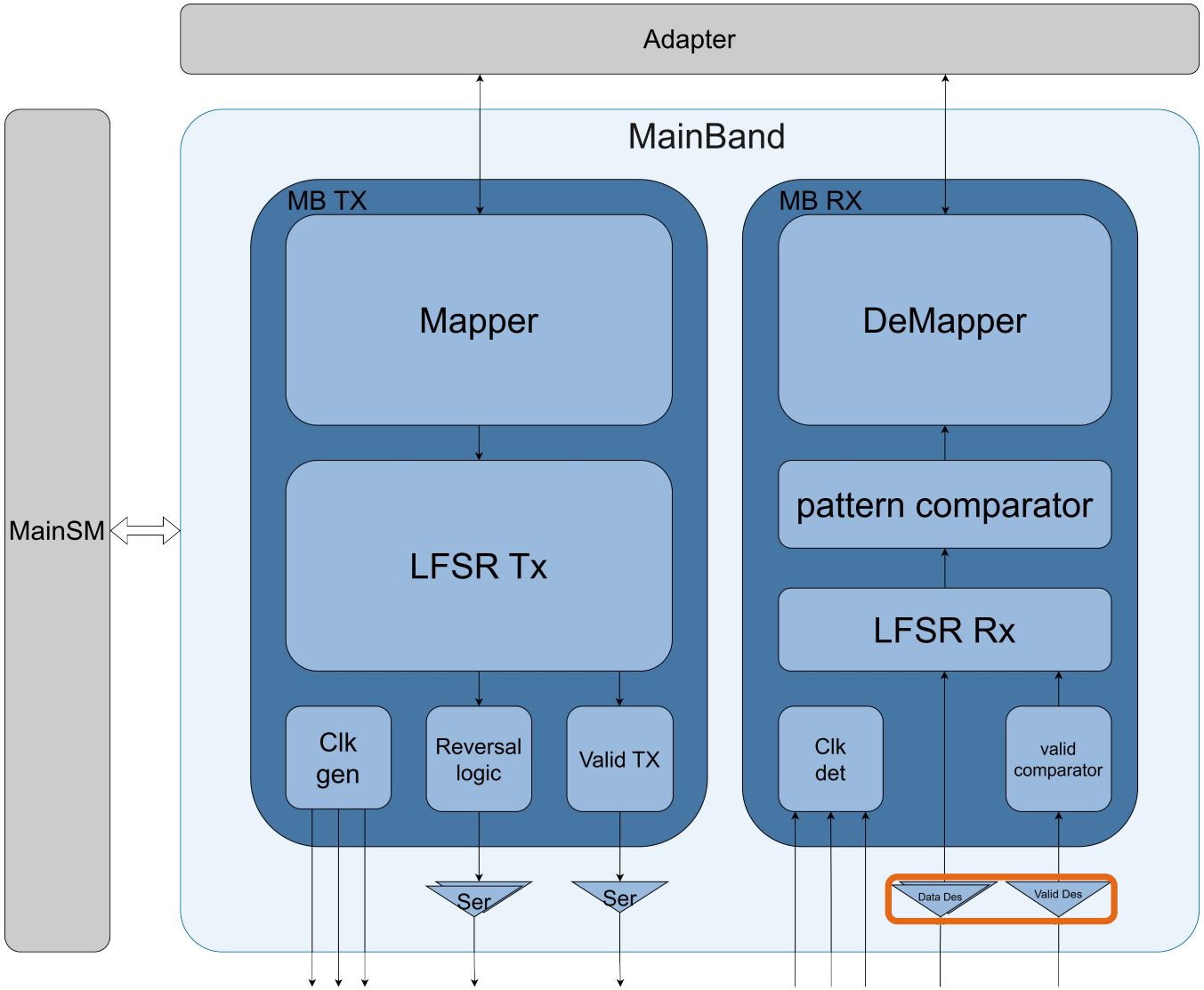

Adapter
MainBand
MainSM
MB TX
Mapper
LFSR Tx
Clk gen
Reversal logic
Valid TX
Ser
Ser
MB RX
DeMapper
pattern comparator
LFSR Rx
Clk det
valid comparator
Data Des
Valid Des

Serializers

- Convert the **lane-parallel data** and valid **streams** into the **high-speed serial outputs**.
- A dedicated **serializer** handles the **Valid signal along side the data lanes.**
- Operate in the forwarded-clock timing domain generated by the clock generation block.

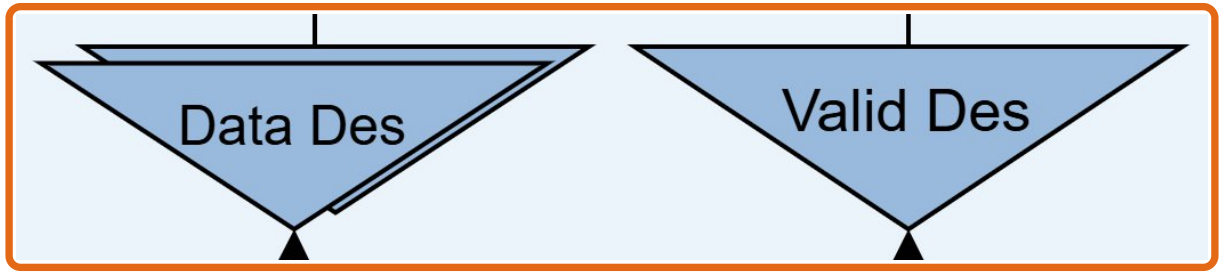


Mainband

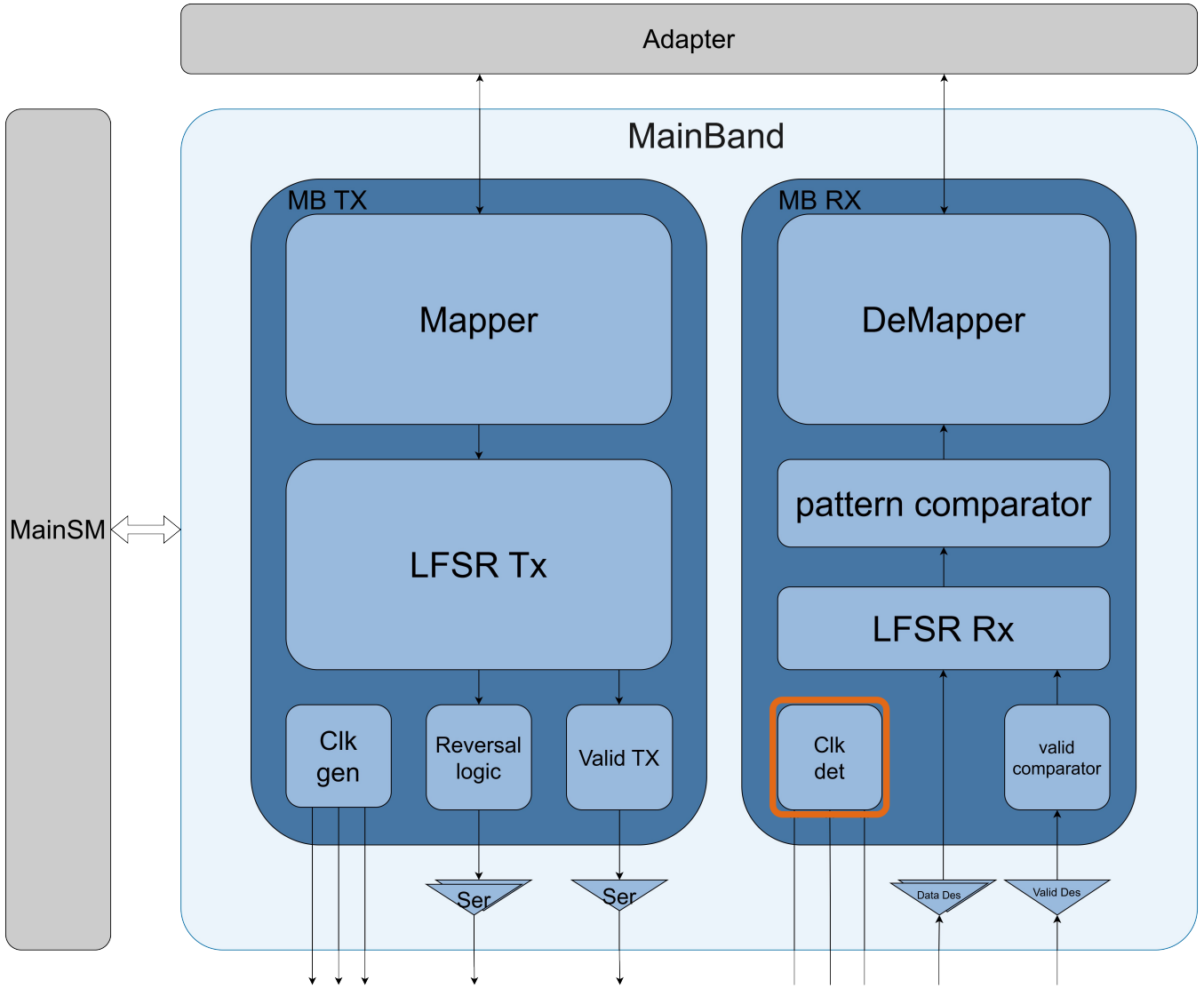

Adapter
MainBand
MainSM
MB TX
Mapper
LFSR Tx
Clk gen
Reversal logic
Valid TX
Ser
Ser
MB RX
DeMapper
pattern comparator
LFSR Rx
Clk det
valid comparator
Data Des
Valid Des

Deserializers

- **Data Des** and **Valid Des** recover **lane-parallel words** from the incoming serial streams.
- Sample the **received data** using the **forwarded clock** from the remote die.
- Feed aligned **parallel data to LFSR RX** and the comparators.

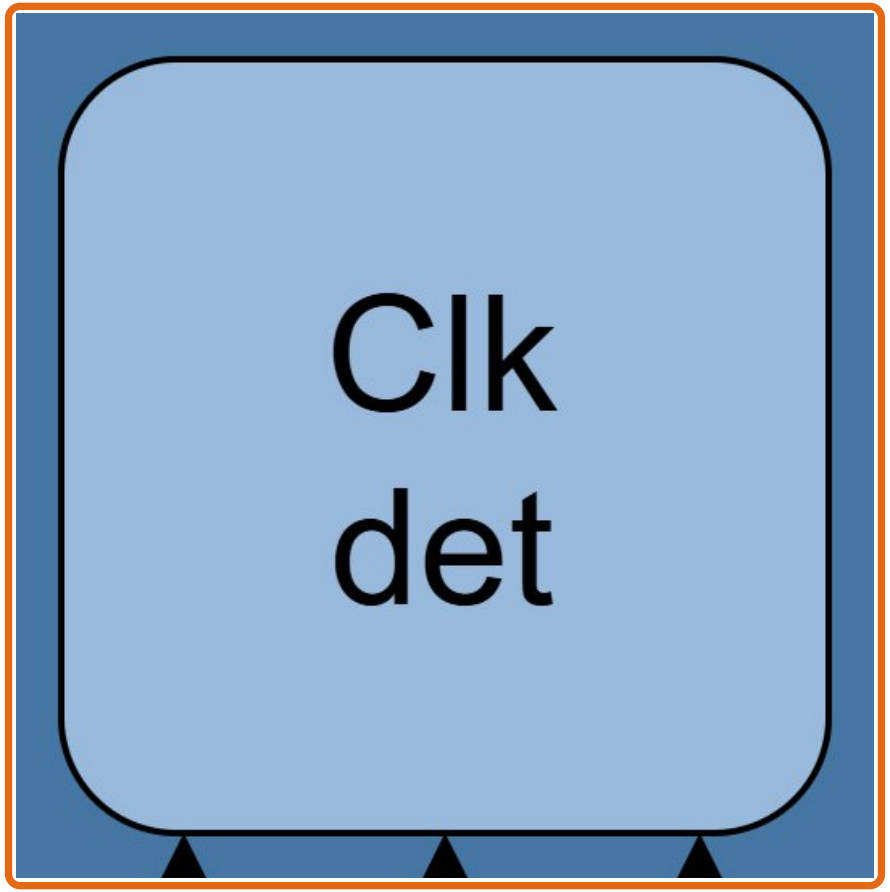


Mainband

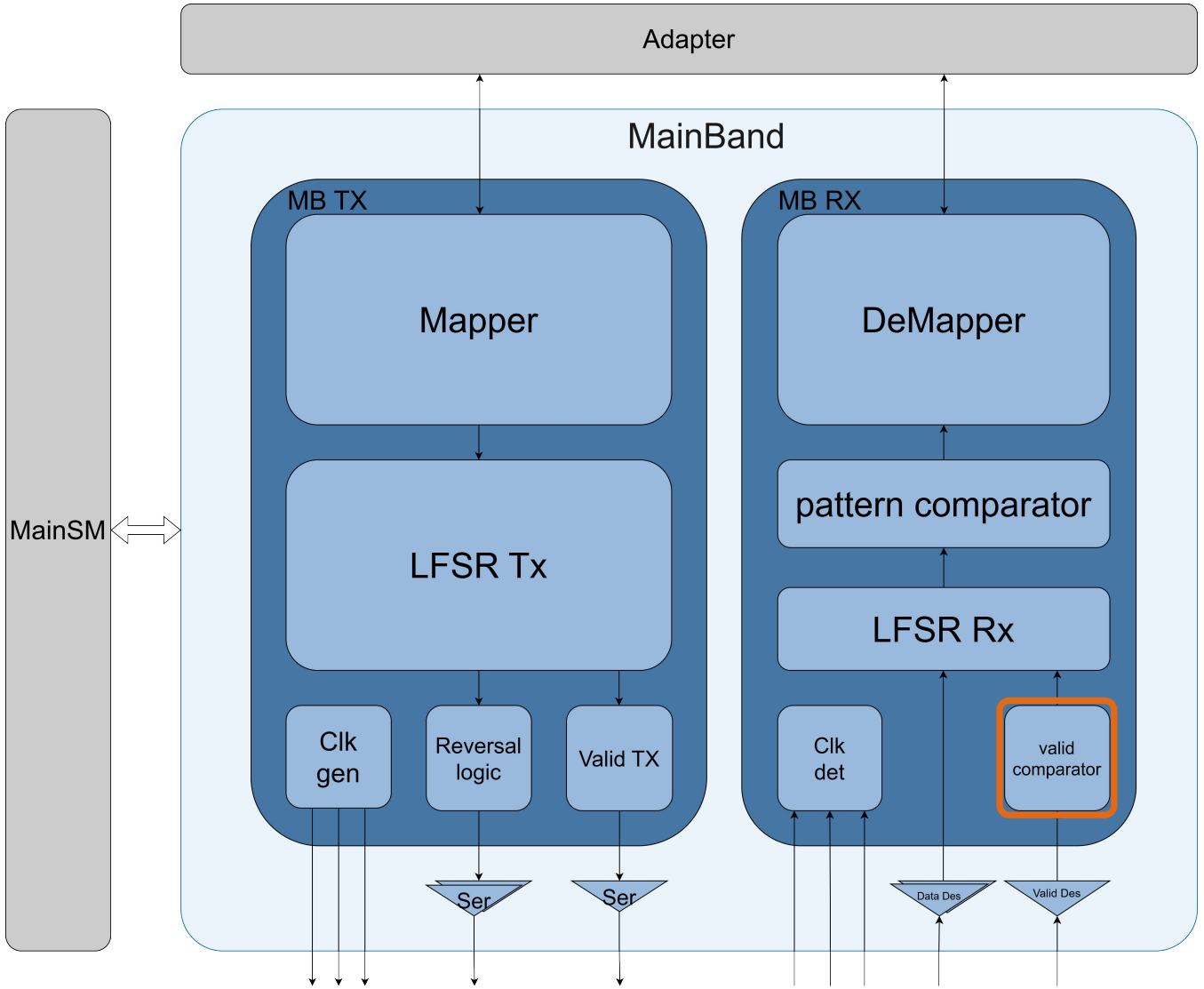

Adapter
MainBand
MainSM
MB TX
Mapper
LFSR Tx
Clk gen
Reversal logic
Valid TX
Ser
Ser
MB RX
DeMapper
pattern comparator
LFSR Rx
Clk det
valid comparator
Data Des
Valid Des

Clock Pattern Detector

- Validates the received **forwarded clock** and **track lane** during early **link training**.
- Checks **each lane** against an expected toggle/quiet pattern structure.
- Asserts sticky **pass flags** once the clock infrastructure is verified ready.

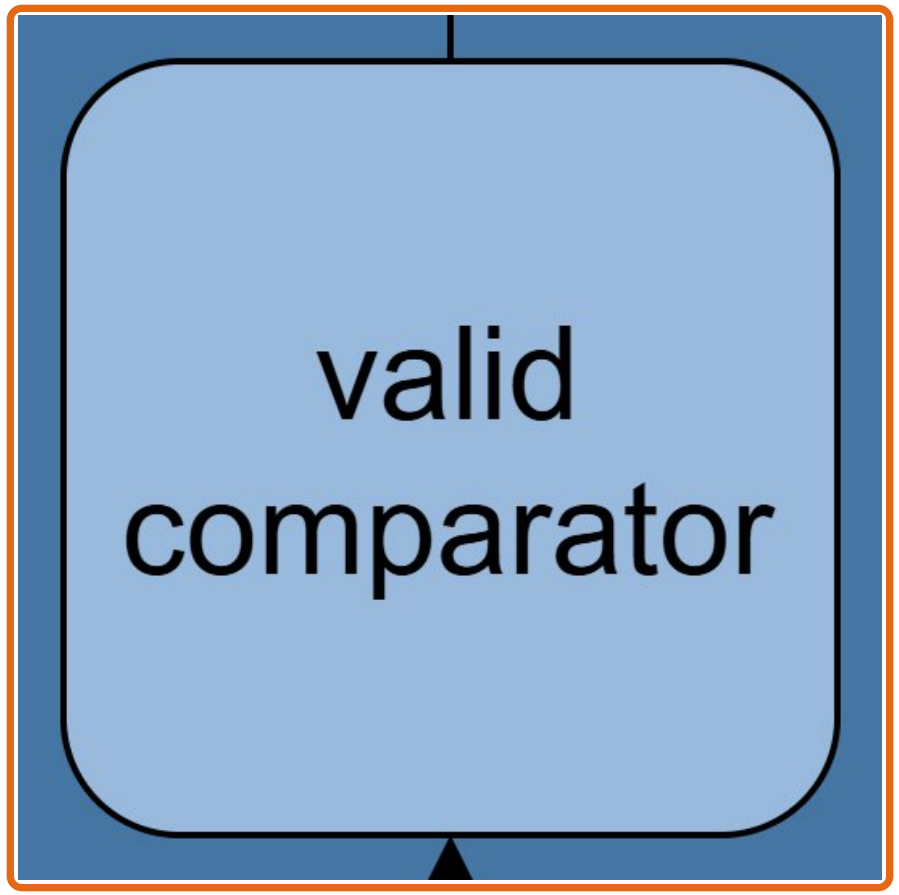


Mainband

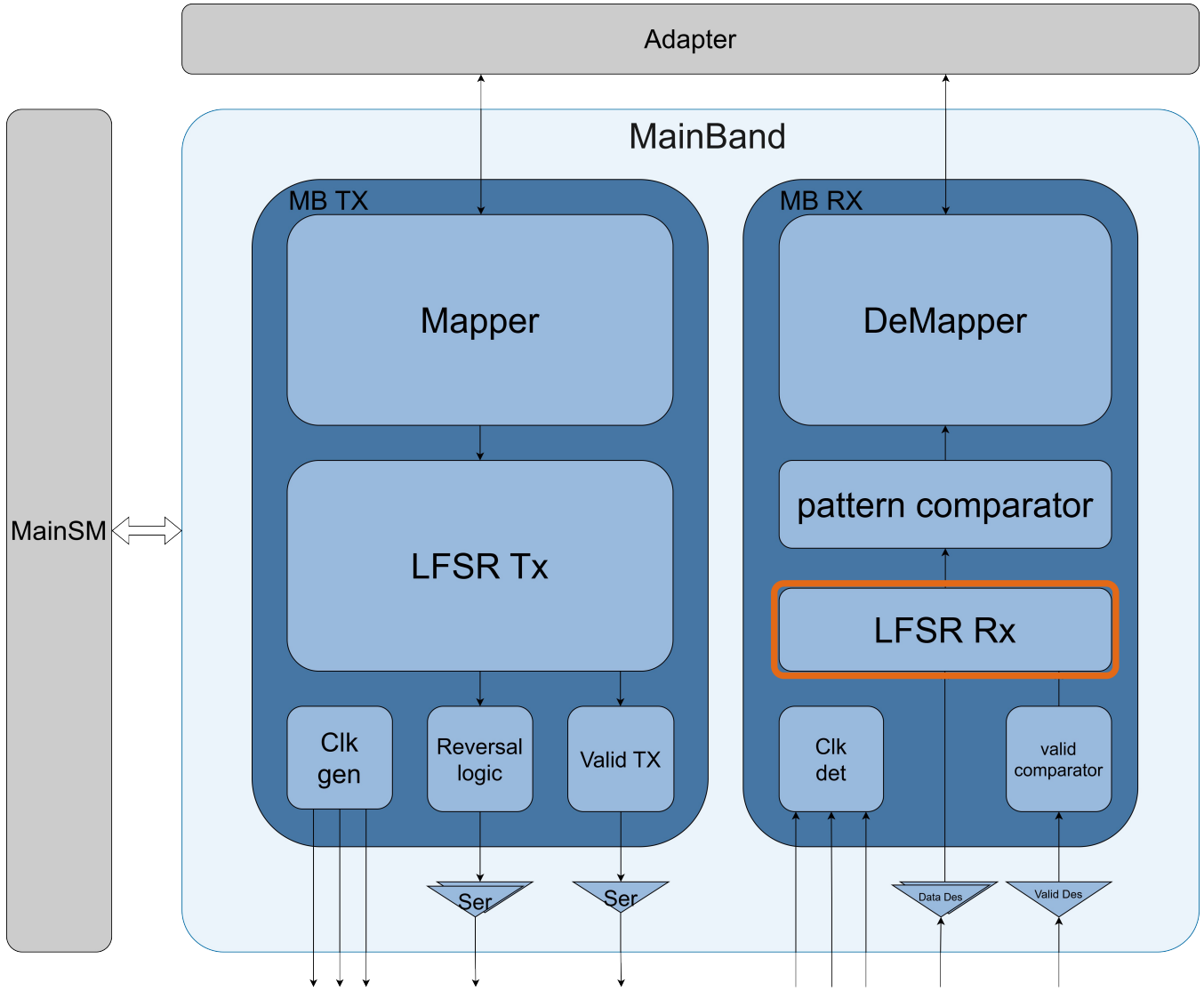

Adapter
MainSM
MainBand
MB TX
Mapper
LFSR Tx
Clk gen
Reversal logic
Valid TX
Ser
Ser
MB RX
DeMapper
pattern comparator
LFSR Rx
Clk det
valid comparator
Data Des
Valid Des

Valid Comparator

- Verifies that the received **Valid pattern matches** the **expected framing structure**.
- Qualifies incoming data bursts before they enter the RX datapath.
- **Reports framing mismatches** to **MainSM** during training.



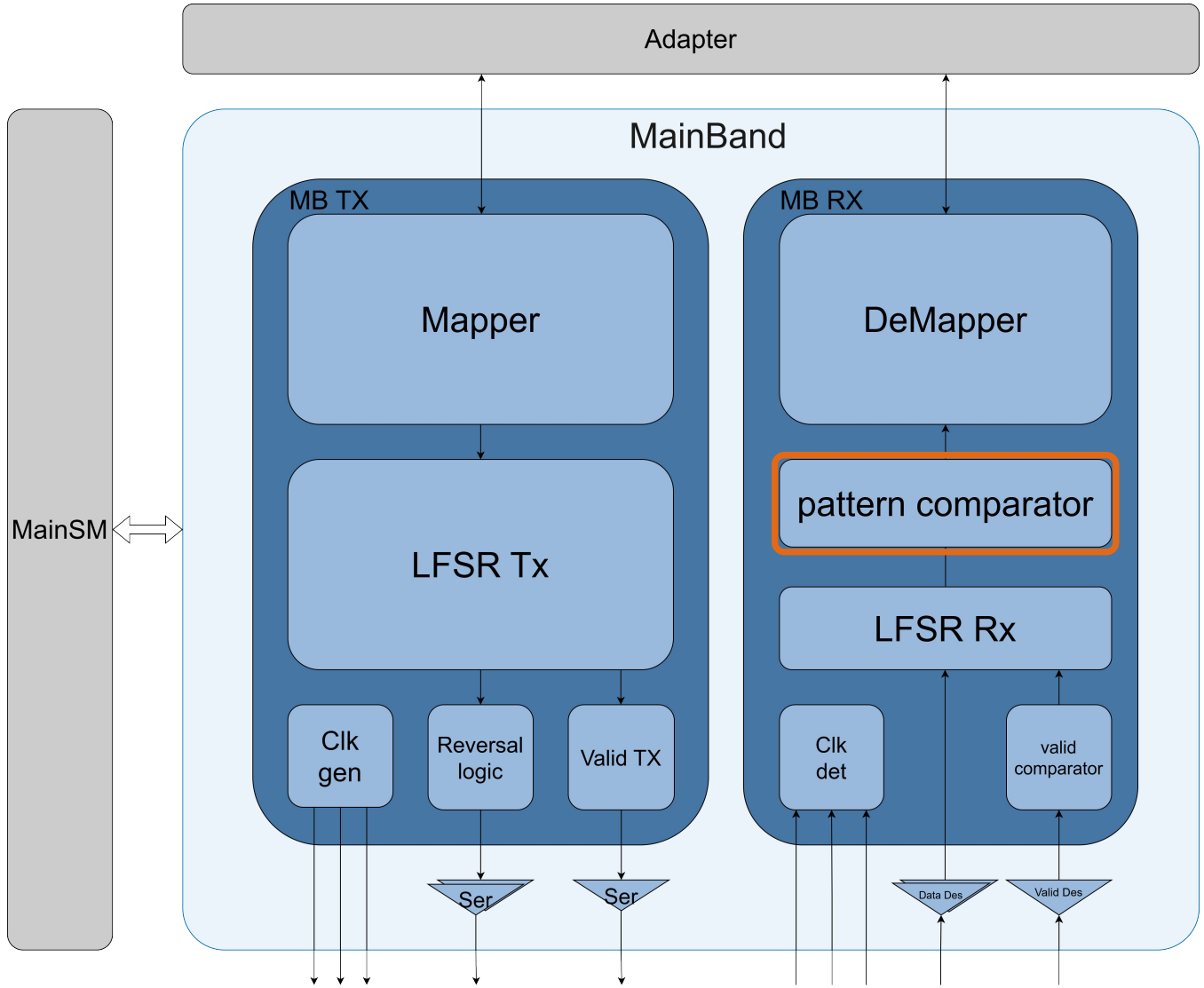
Mainband


Adapter
MainSM
MainBand
MB TX
Mapper
LFSR Tx
Clk gen
Reversal logic
Valid TX
Ser
Ser
MB RX
DeMapper
pattern comparator
LFSR Rx
Clk det
valid comparator
Data Des
Valid Des

LFSR RX

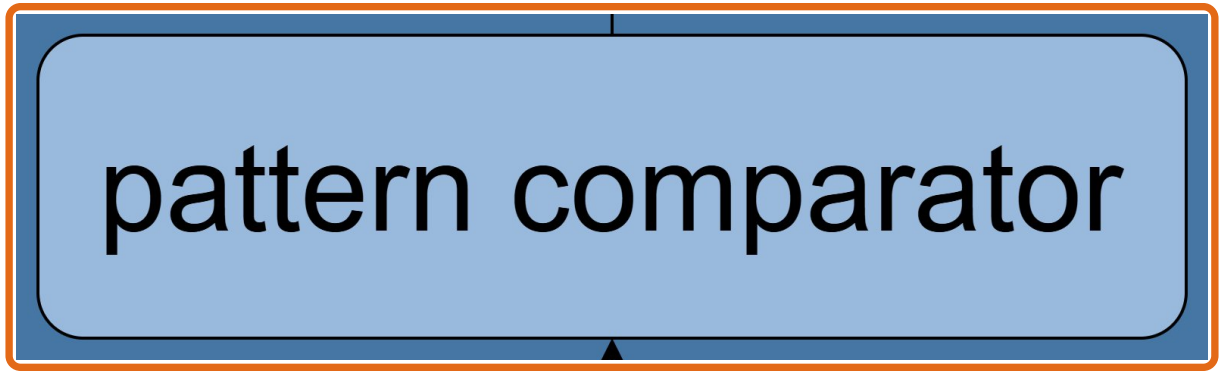
- Performs the **inverse LFSR operation** of **LFSR_TX** to **descramble** incoming payload data.
- Generates local reference **PRBS-23/Per-Lane ID** sequences for training comparison.
- Adapts to **link width (x16/x8/x4)** and forwards **descrambled data** to the Demapper.

Mainband

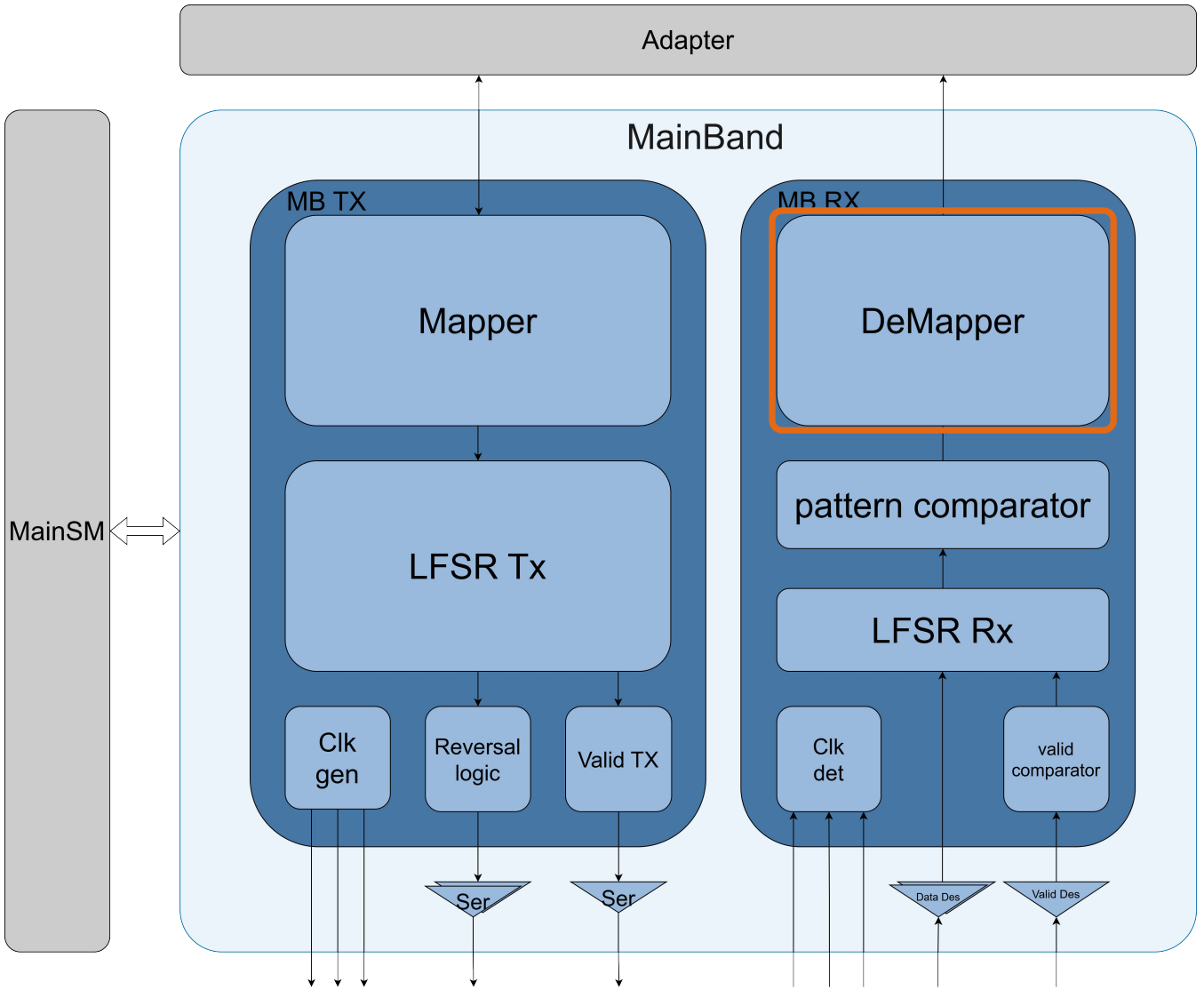

Adapter
MainBand
MainSM
MB TX
Mapper
LFSR Tx
Clk gen
Reversal logic
Valid TX
Ser
Ser
MB RX
DeMapper
pattern comparator
LFSR Rx
Clk det
valid comparator
Data Des
Valid Des

Pattern Comparator

- Cross-**compares received training data** against the local **LFSR/Per-Lane ID** reference.
- Supports **Per-Lane mode** (individual lane pass/fail) and **Aggregate mode** (link-wide error count).
- Reports **pass/fail** results and test completion to qualify lane health.

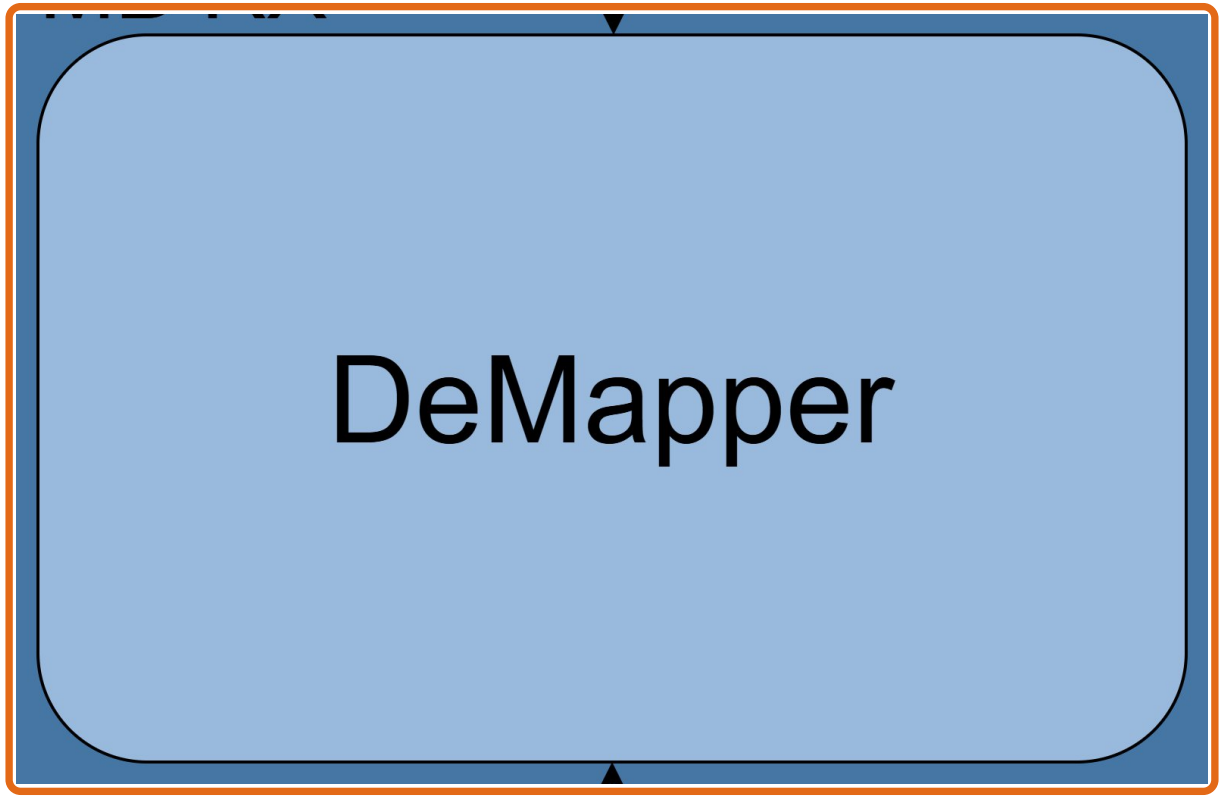


Mainband

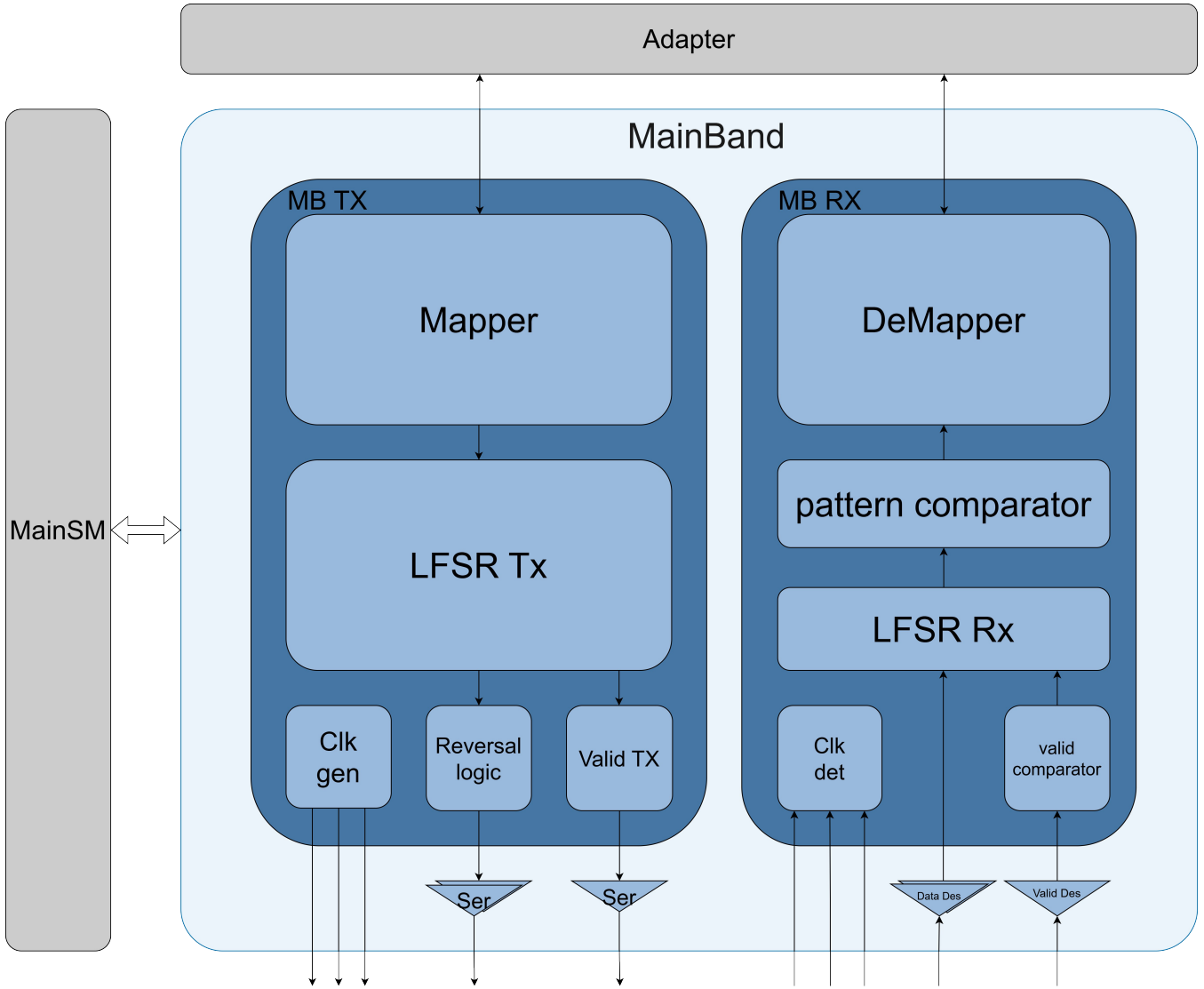

Adapter
MainBand
MainSM
MB TX
Mapper
LFSR Tx
Clk gen
Reversal logic
Valid TX
Ser
Ser
MB RX
DeMapper
pattern comparator
LFSR Rx
Clk det
valid comparator
Data Des
Valid Des

Demapper

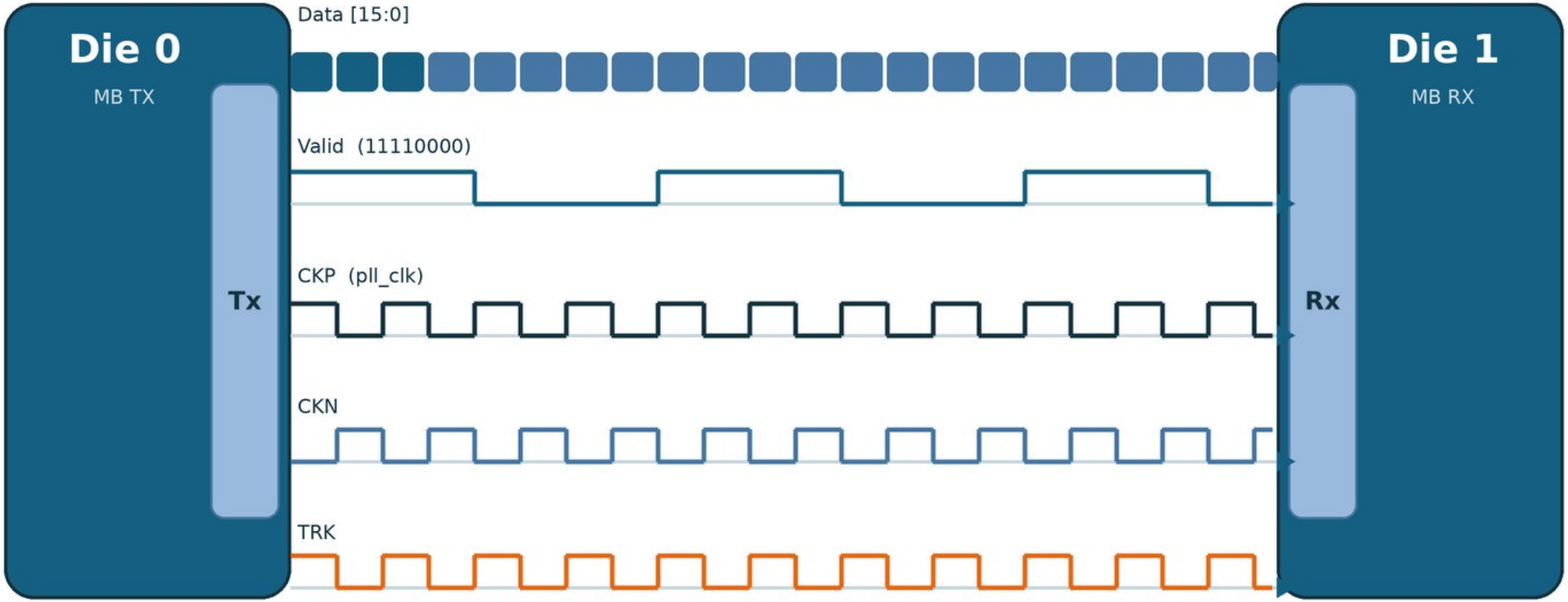
- **Reverses the Mapper's operation**, merging 16 lane streams back into a **512-bit word.**
- Reassembles data byte-by-byte, **adapting cycle count** to the **active link width.**
- Asserts a **valid flag** once the full output word is **reconstructed.**



Mainband

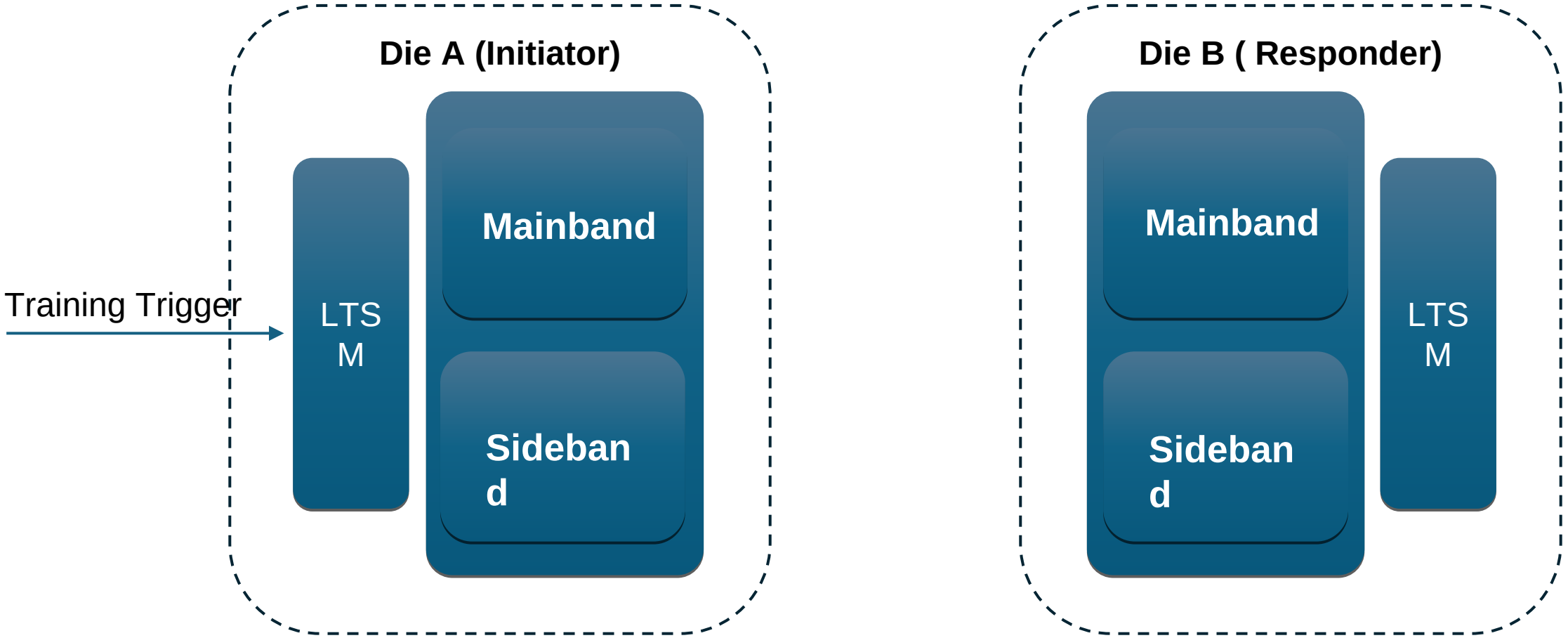

Adapter
MainSM
MainBand
MB TX
Mapper
LFSR Tx
Clk gen
Reversal logic
Valid TX
Ser
Ser
MB RX
DeMapper
pattern comparator
LFSR Rx
Clk det
valid comparator
Data Des
Valid Des

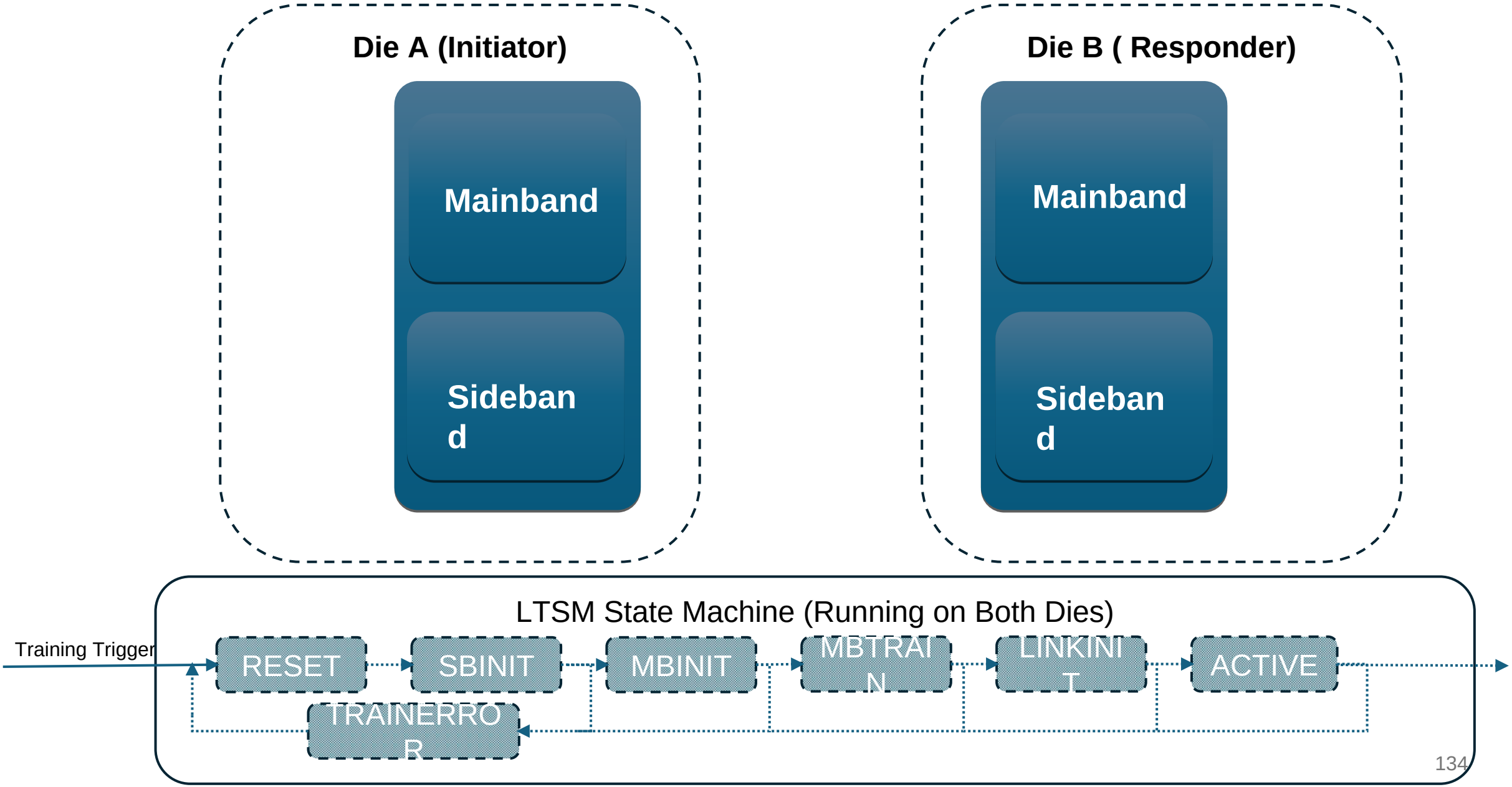
Mainband Interface

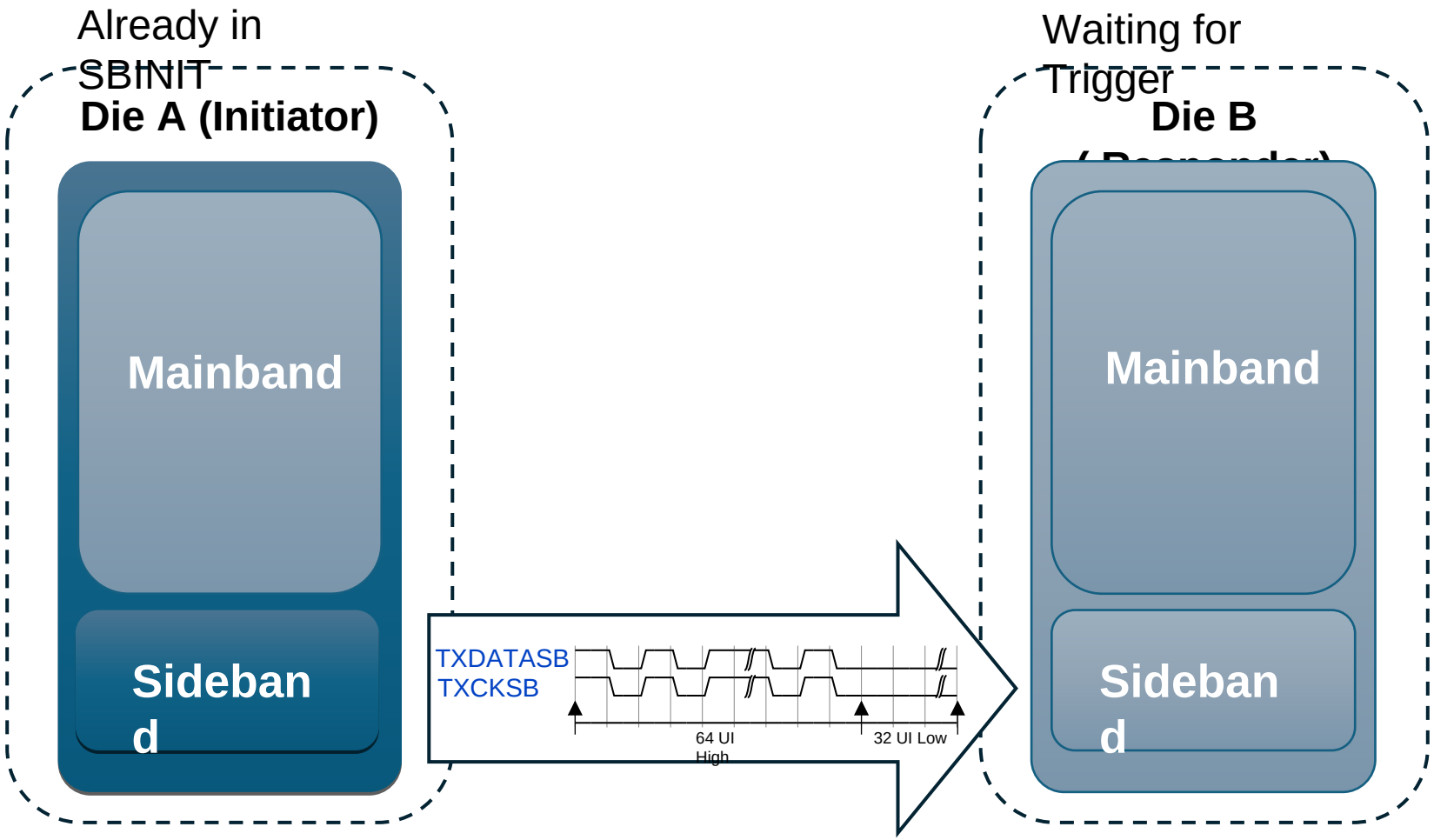


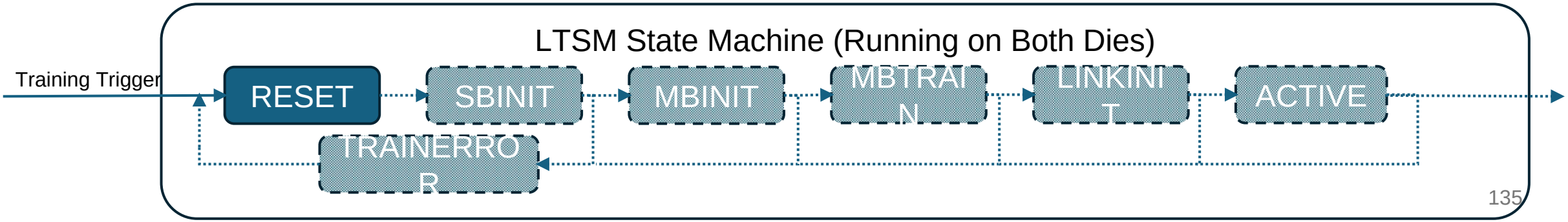
pll_clk is DDR → 2 bits serialized per clock cycle · Valid = 4 UI high / 4 UI low

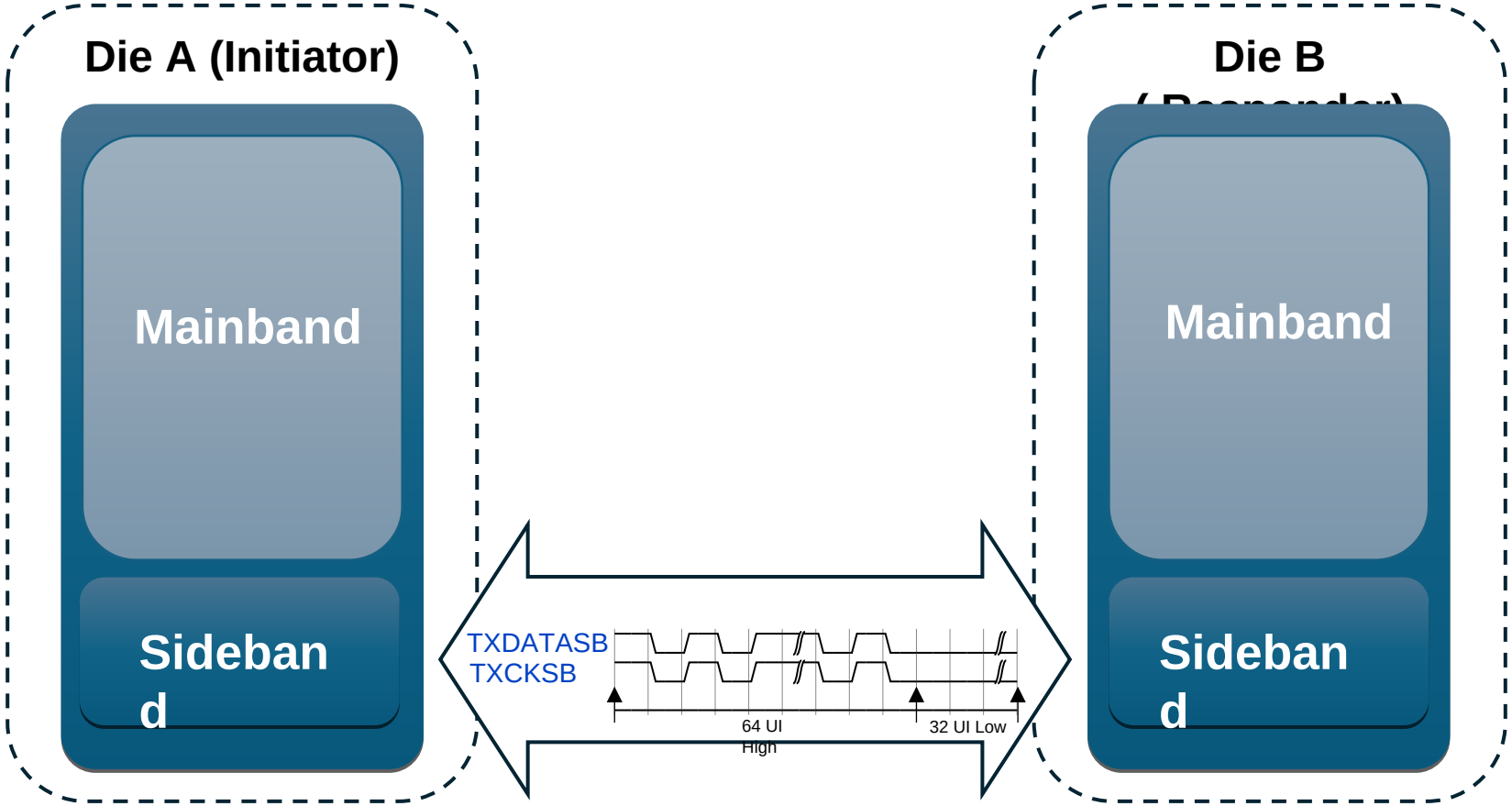
Bring Up Flow

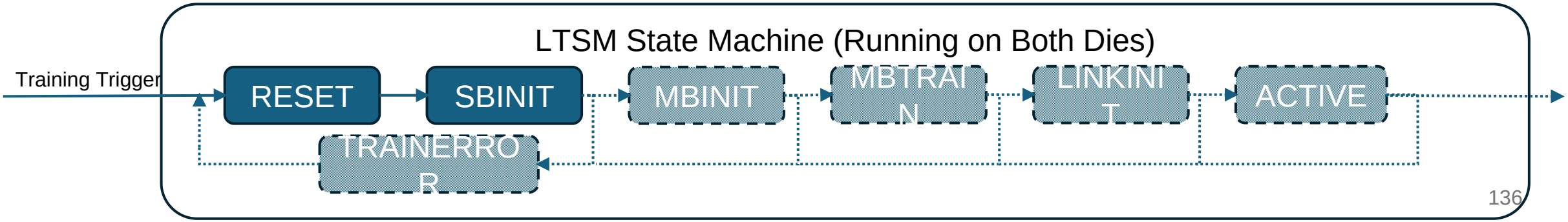

Die A (Initiator)
Training Trigger
LTS
M
Mainband
Sideban
d
Die B (Responder)
Mainband
Sideban
d
LTS
M

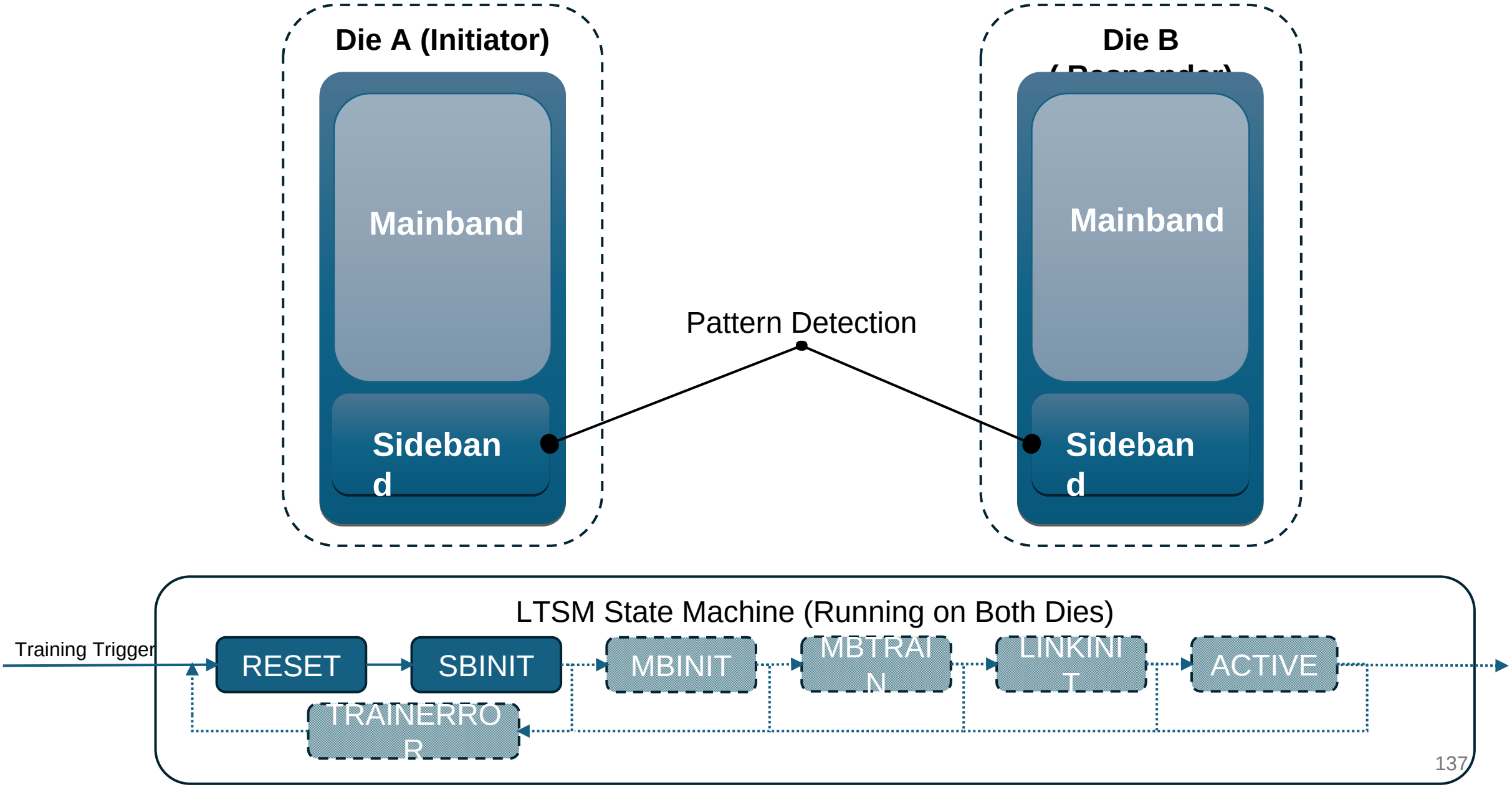

Die A (Initiator)
Mainband
Sideband
Die B (Responder)
Mainband
Sideband
LTSM State Machine (Running on Both Dies)
Training Trigger
RESET
SBINIT
MBINIT
MBTRAIN
LINKINIT
ACTIVE
TRAINERROR

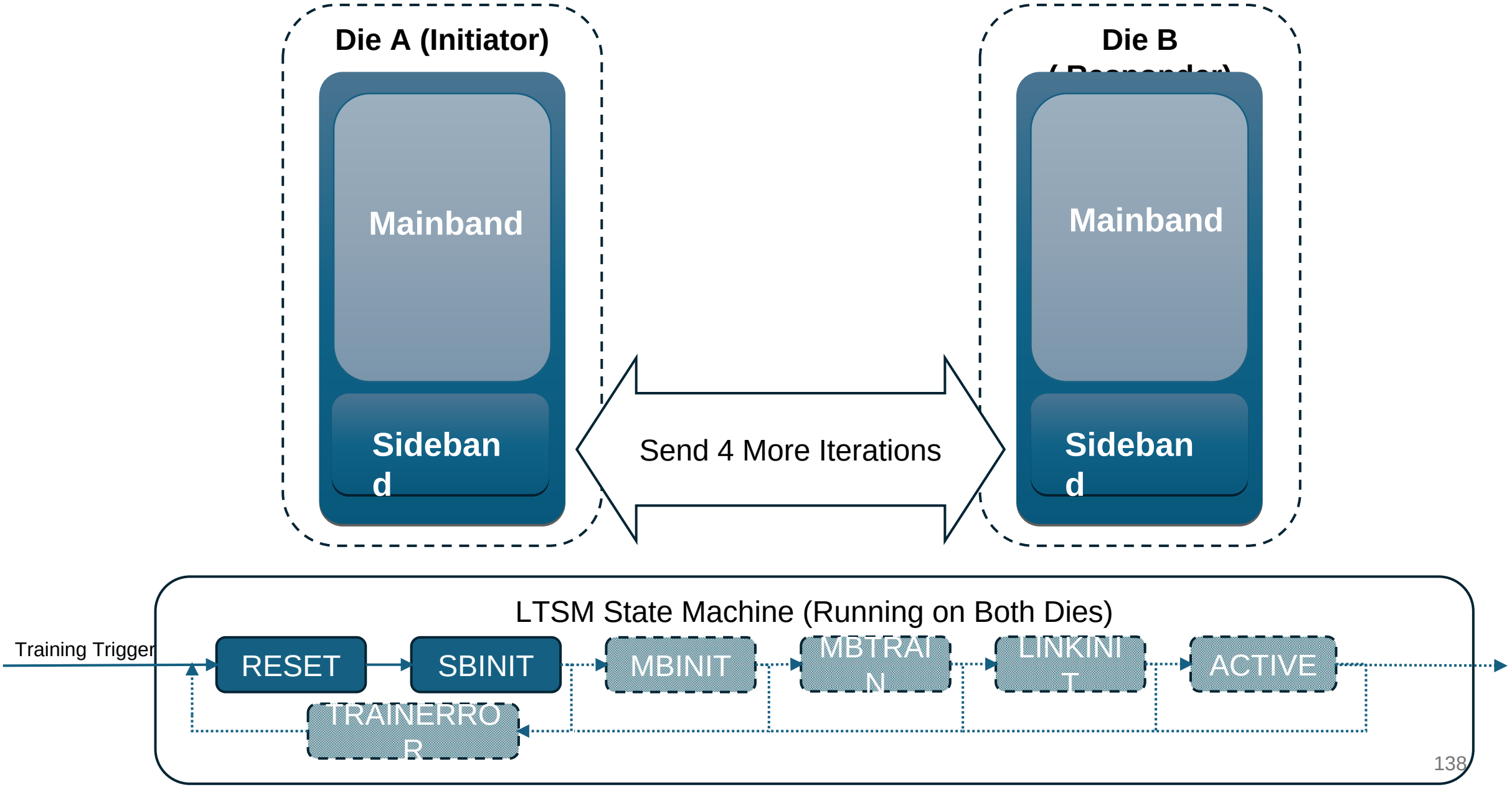

Already in SBINIT
Die A (Initiator)
Mainband
Sideband
TXDATASB
TXCKSB
64 UI High
32 UI Low
Waiting for Trigger
Die B
Mainband
Sideband

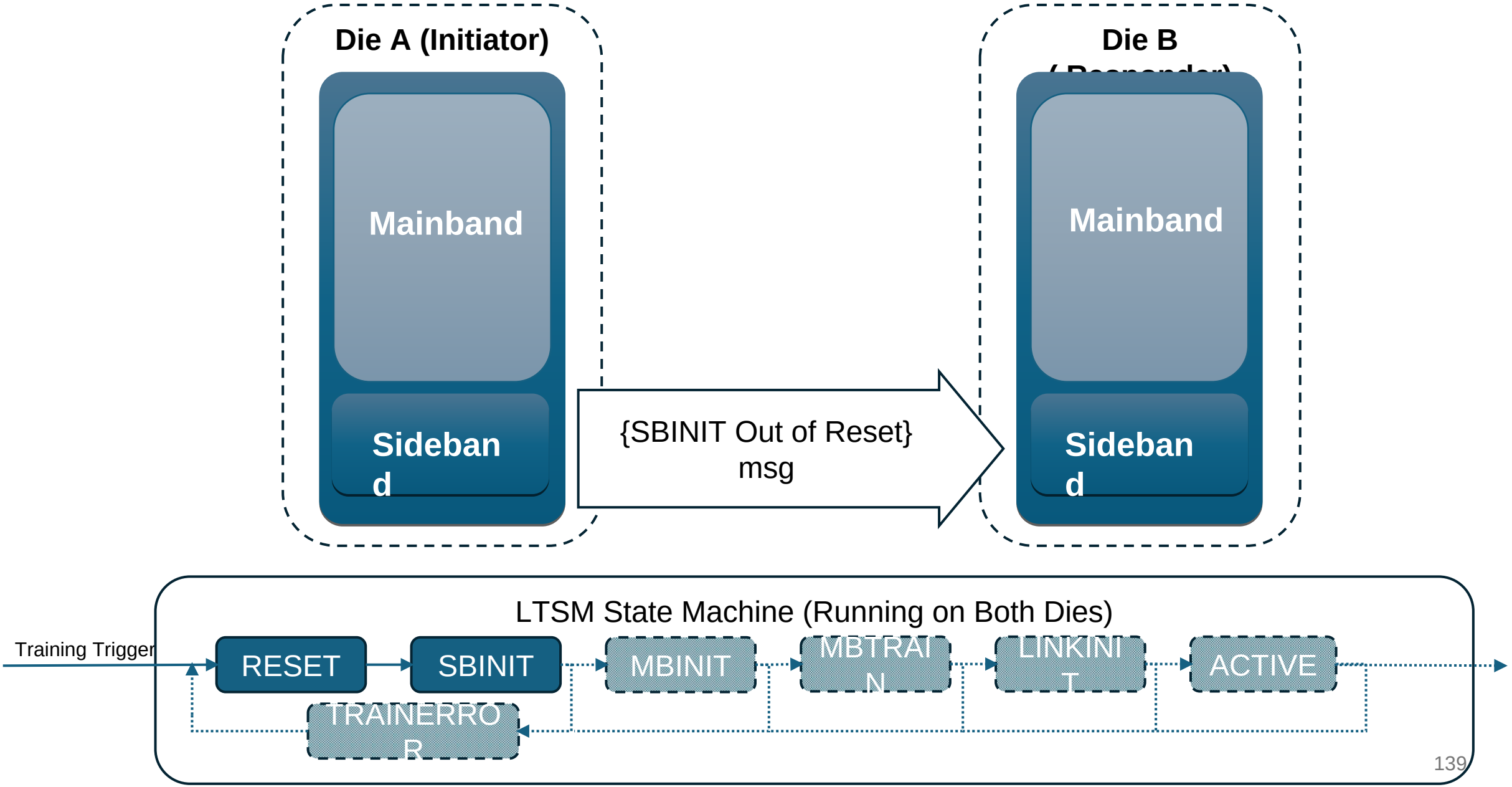

LTSM State Machine (Running on Both Dies)
Training Trigger
RESET
SBINIT
MBINIT
MBTRAIN
LINKINIT
ACTIVE
TRAINERROR

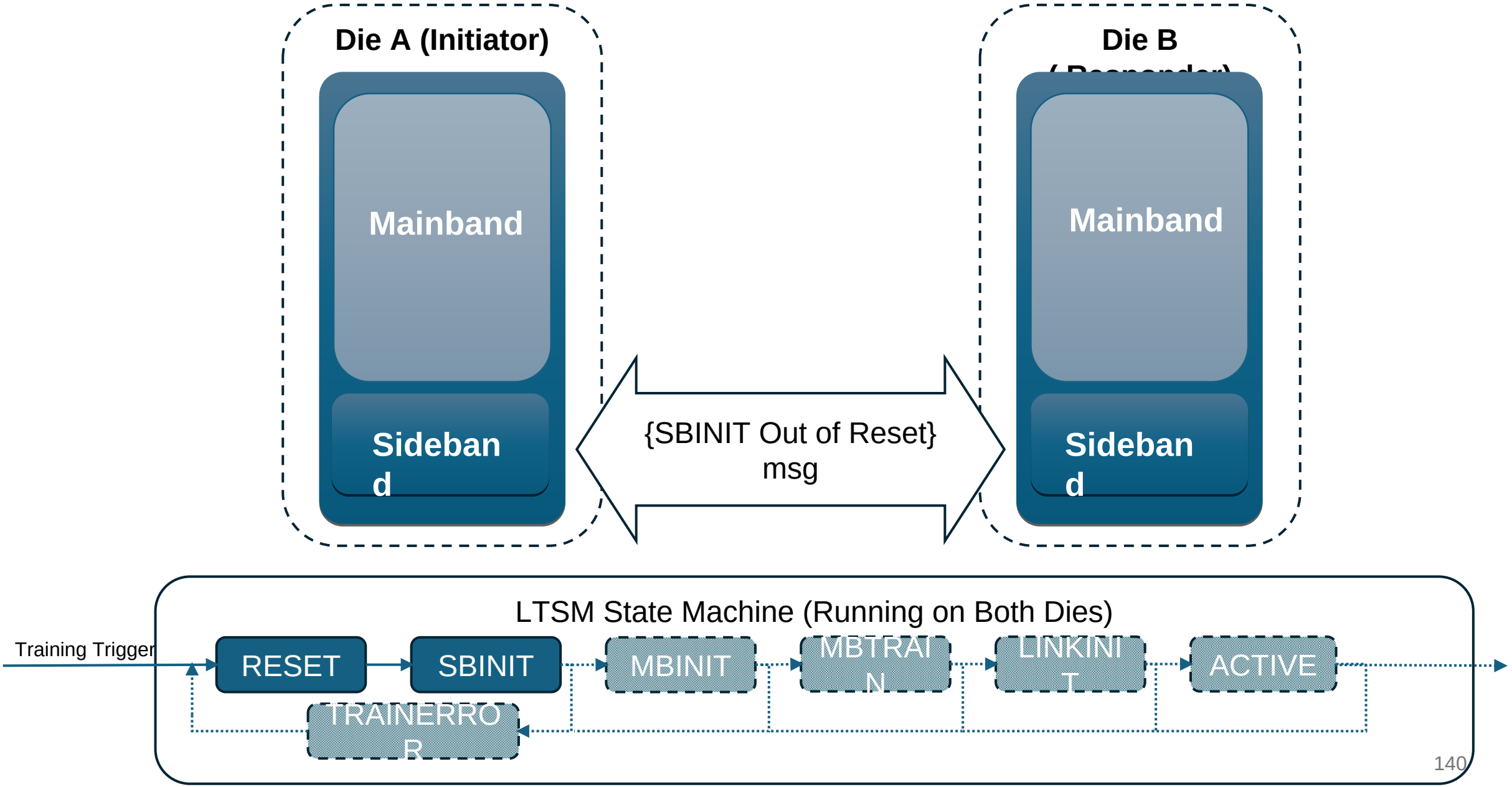

Die A (Initiator)
Mainband
Sideband
Die B
Mainband
Sideband
TXDATASB
TXCKSB
64 UI
High
32 UI Low

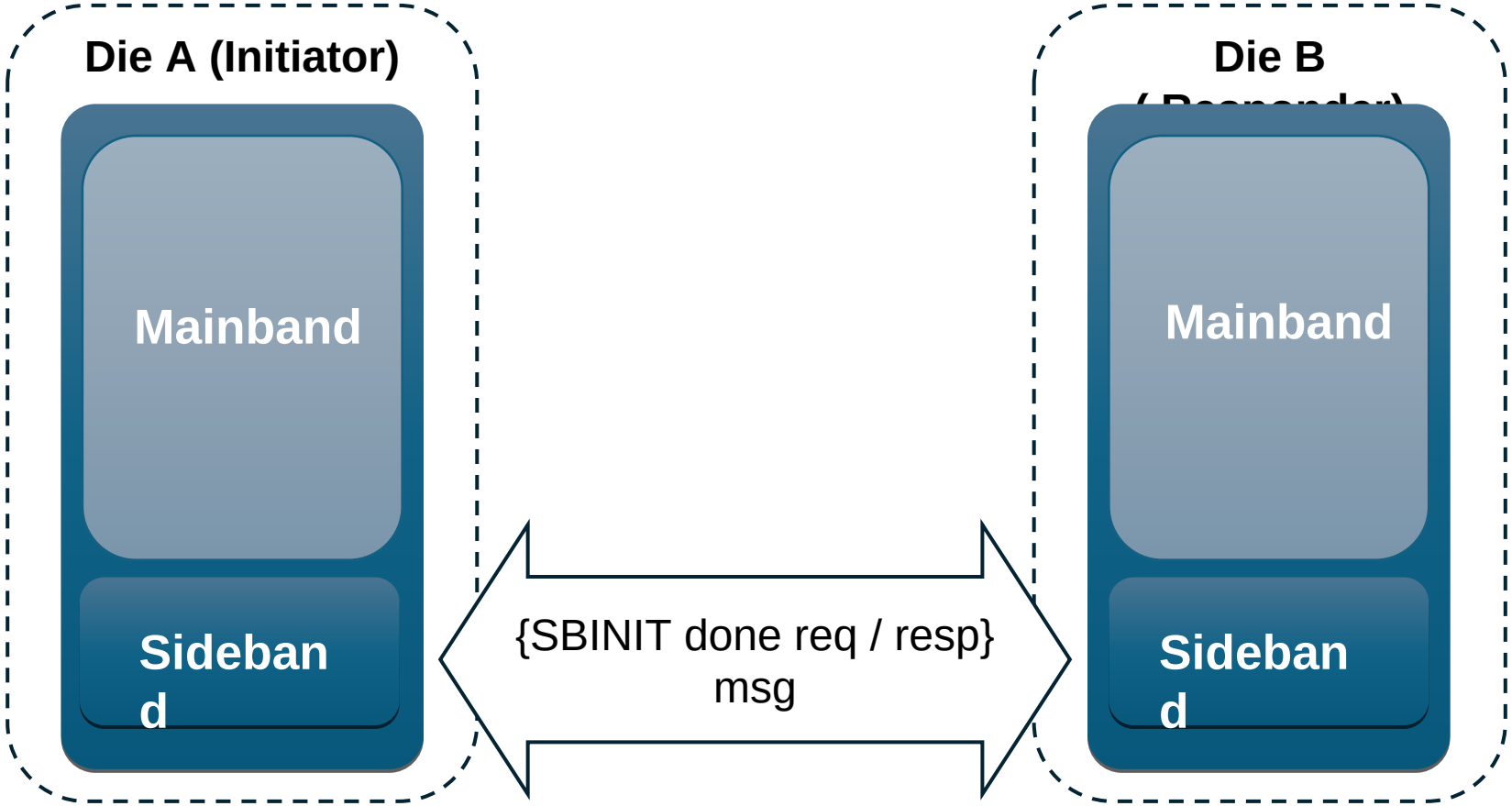

LTSM State Machine (Running on Both Dies)
Training Trigger
RESET
SBINIT
MBINIT
MBTRAIN
LINKINIT
ACTIVE
TRAINERROR

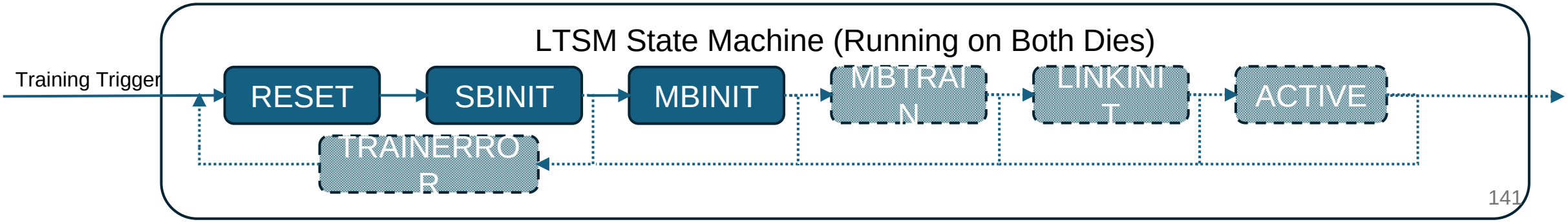

Die A (Initiator)
Mainband
Sideband
Pattern Detection
Die B
Mainband
Sideband
LTSM State Machine (Running on Both Dies)
Training Trigger
RESET
SBINIT
MBINIT
MBTRAIN
LINKINIT
ACTIVE
TRAINERROR

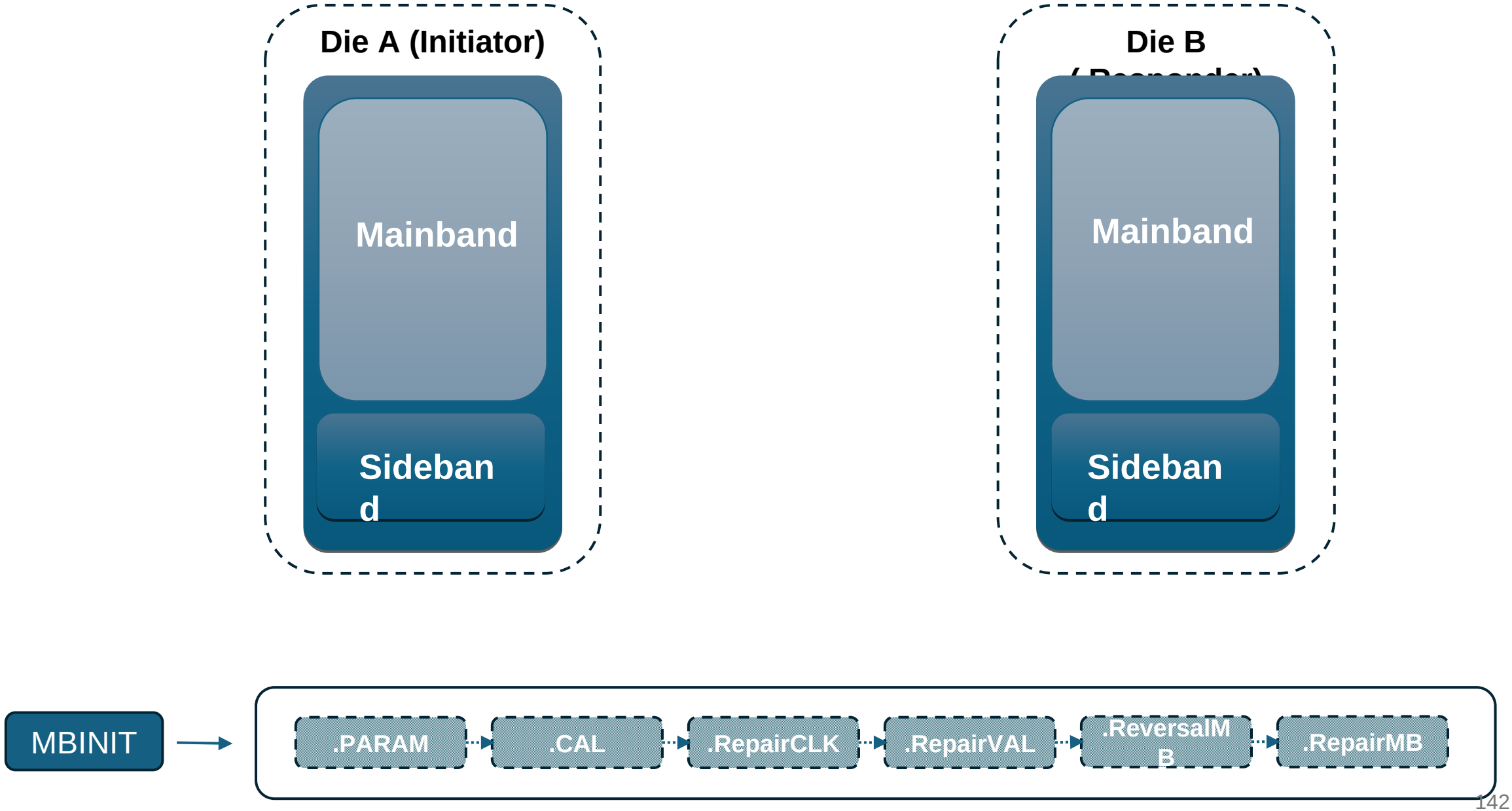

Die A (Initiator)
Mainband
Sideband
Send 4 More Iterations
Die B
Mainband
Sideband
LTSM State Machine (Running on Both Dies)
Training Trigger
RESET
SBINIT
MBINIT
MBTRAIN
LINKINIT
ACTIVE
TRAINERROR

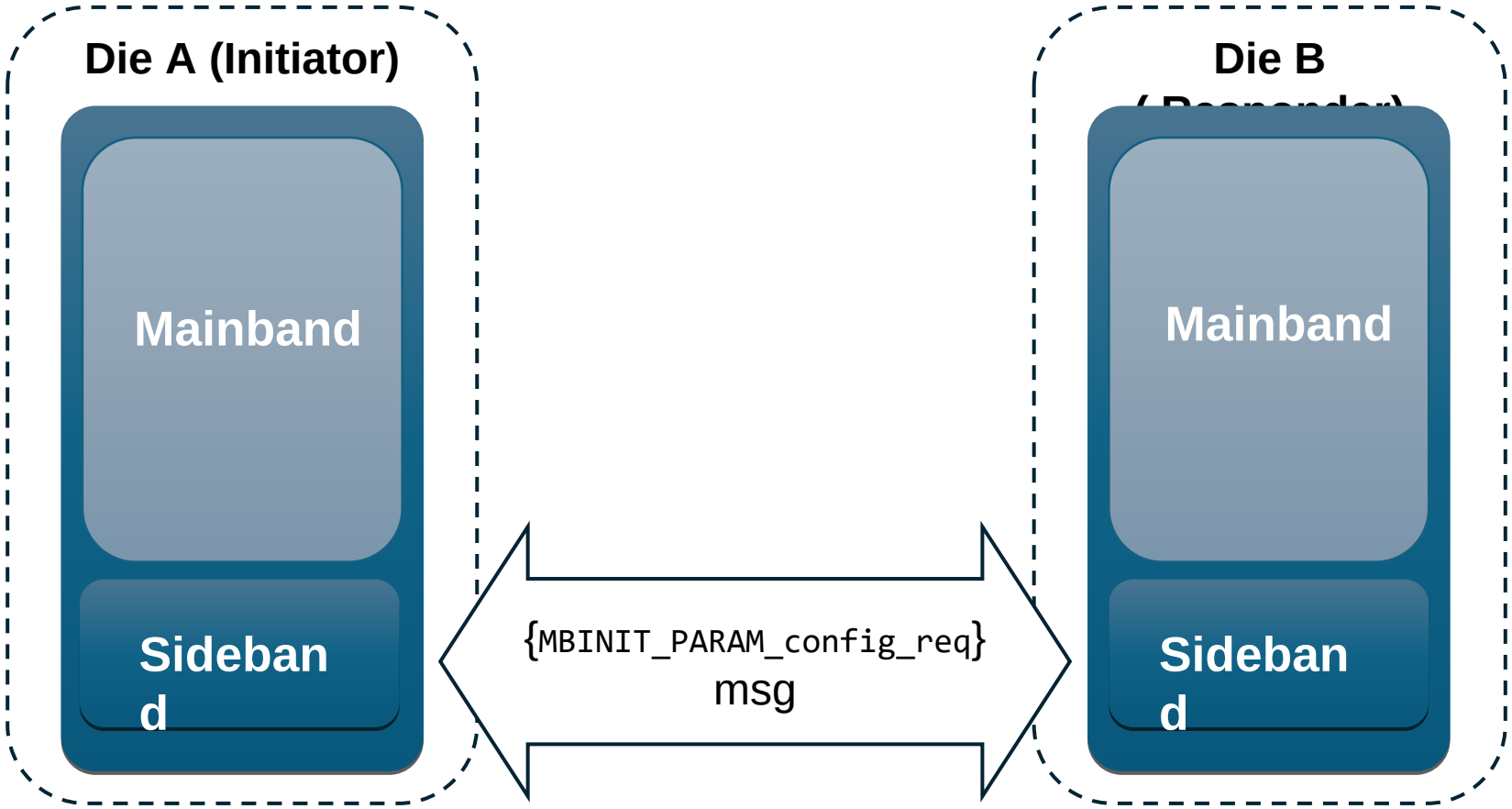

Die A (Initiator)
Mainband
Sideband
{SBINIT Out of Reset} msg
Die B
Mainband
Sideband
LTSM State Machine (Running on Both Dies)
Training Trigger
RESET
SBINIT
MBINIT
MBTRAIN
LINKINIT
ACTIVE
TRAINERROR

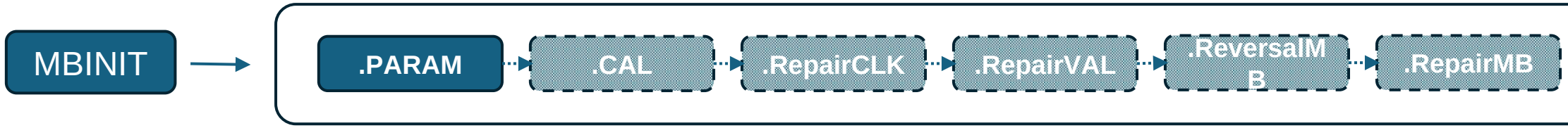

Die A (Initiator)
Mainband
Sideband
{SBINIT Out of Reset} msg
Die B
Mainband
Sideband
LTSM State Machine (Running on Both Dies)
Training Trigger
RESET
SBINIT
MBINIT
MBTRAIN
LINKINIT
ACTIVE
TRAINERROR

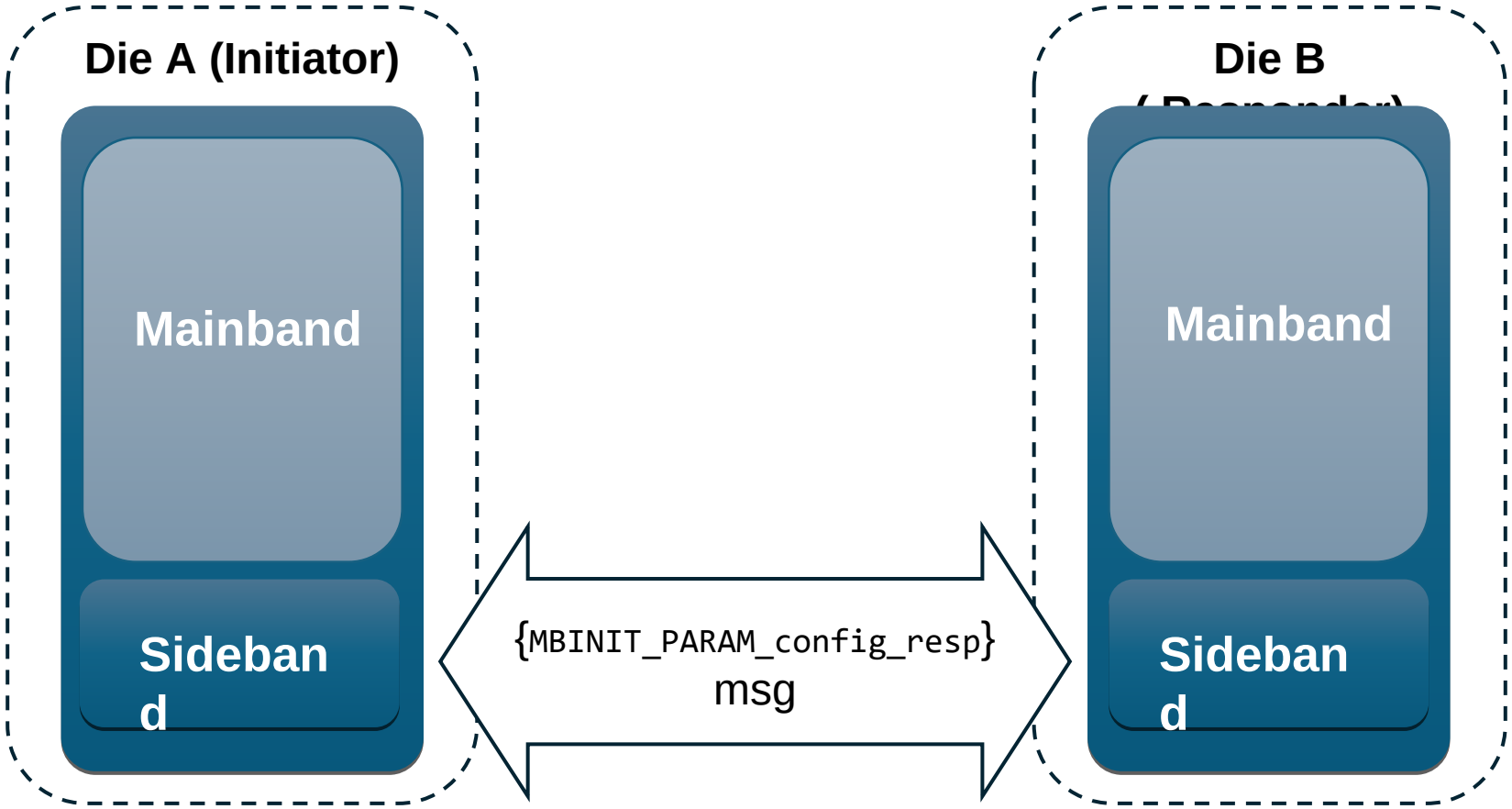

Die A (Initiator)
Mainband
Sideband
{SBINIT done req / resp} msg
Die B
Mainband
Sideband

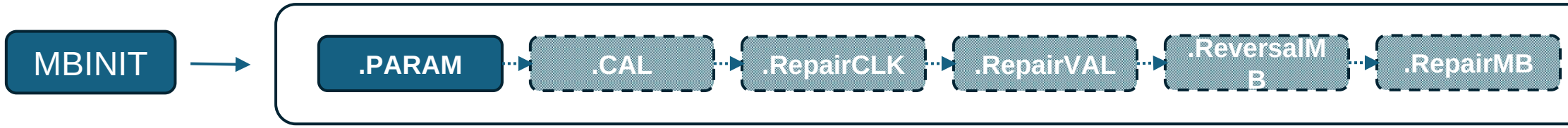

LTSM State Machine (Running on Both Dies)
Training Trigger
RESET
SBINIT
MBINIT
MBTRAIN
LINKINIT
ACTIVE
TRAINERROR

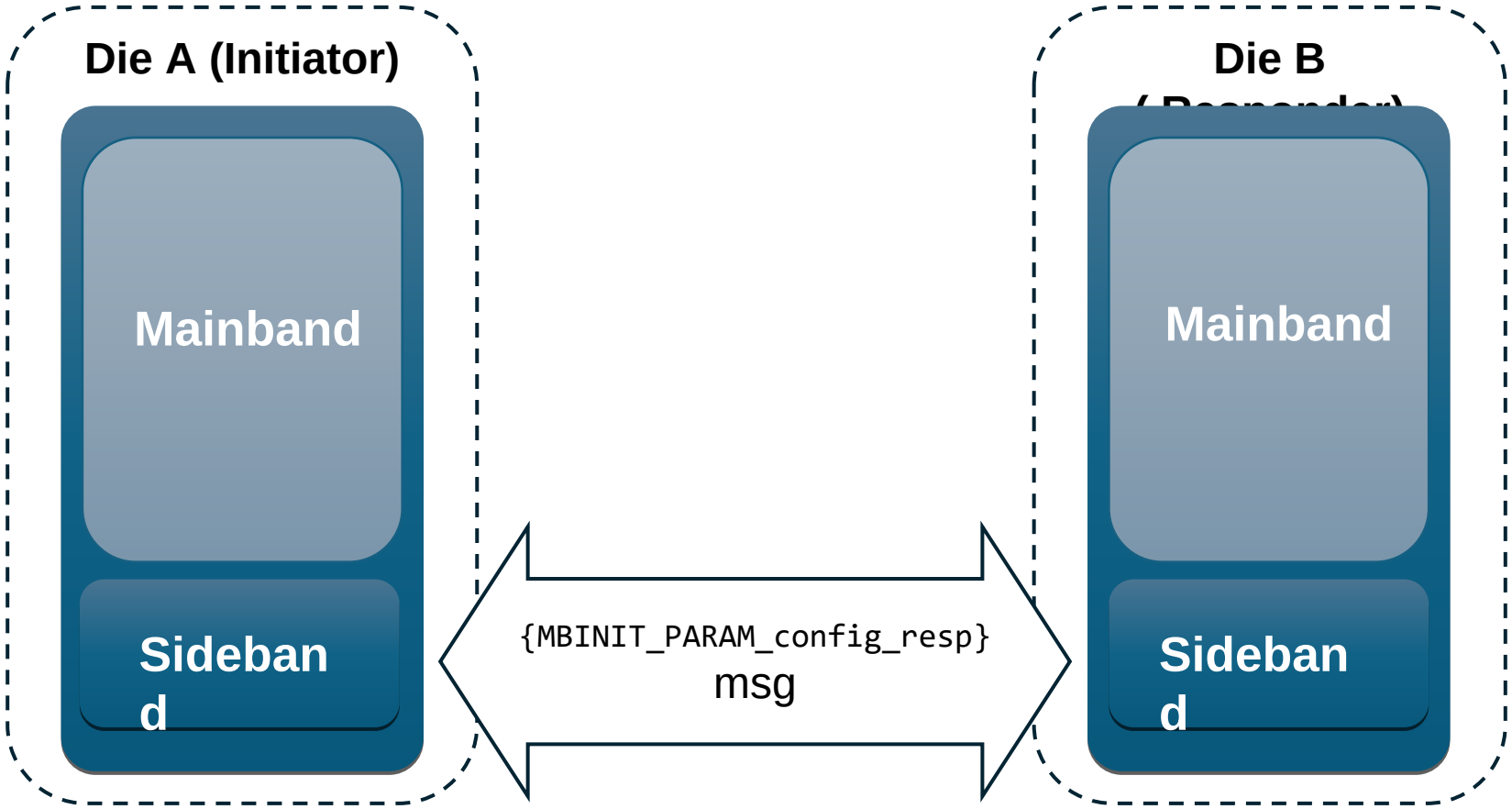

Die A (Initiator)
Mainband
Sideband
Die B
Mainband
Sideband
MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

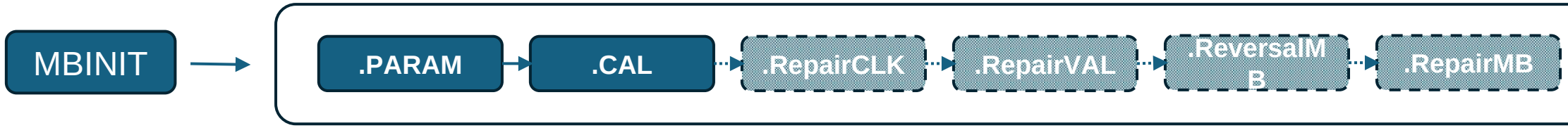

Die A (Initiator)
Mainband
Sideband
{MBINIT_PARAM_config_req}
msg
Die B
Mainband
Sideband

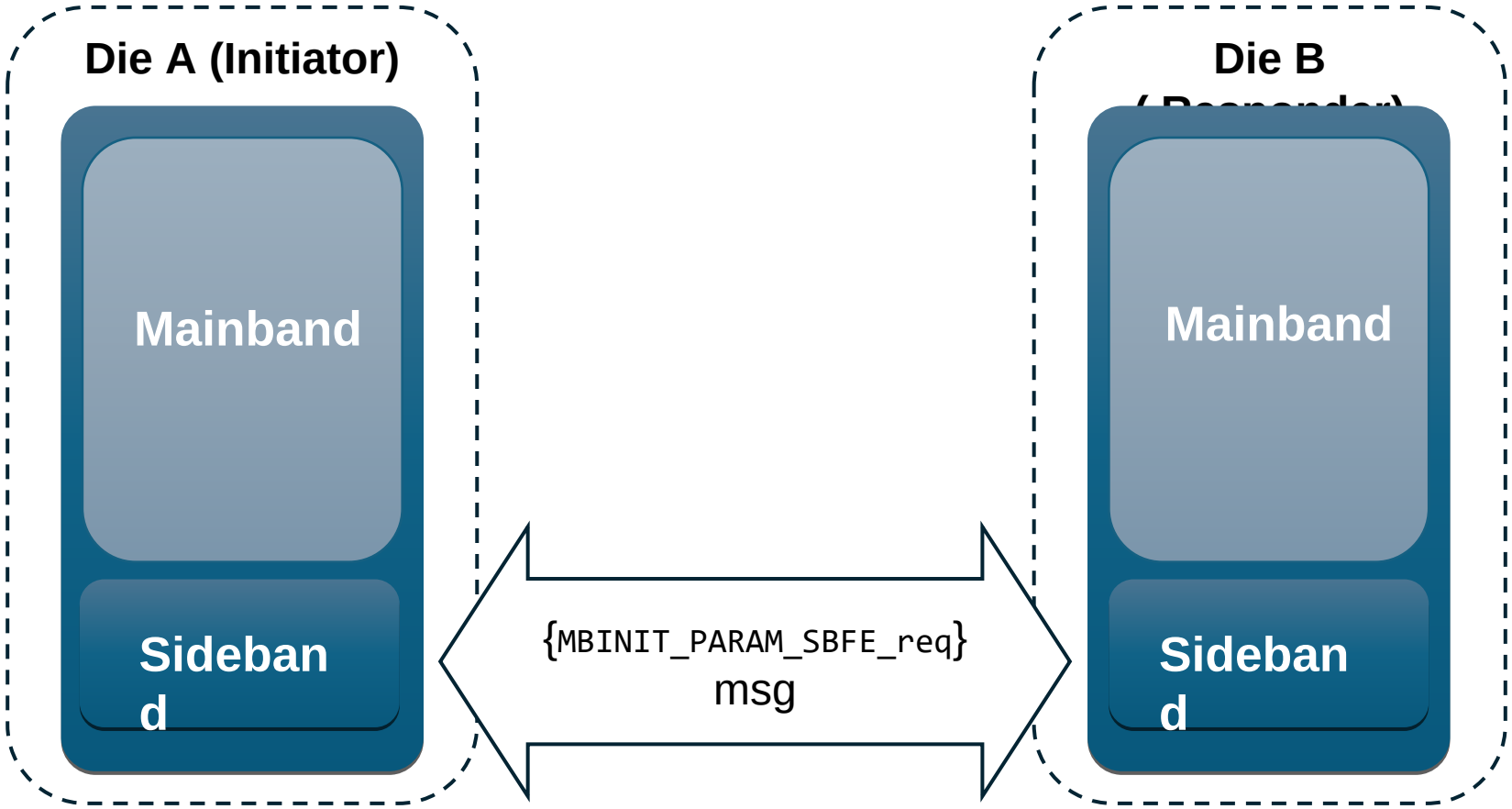

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

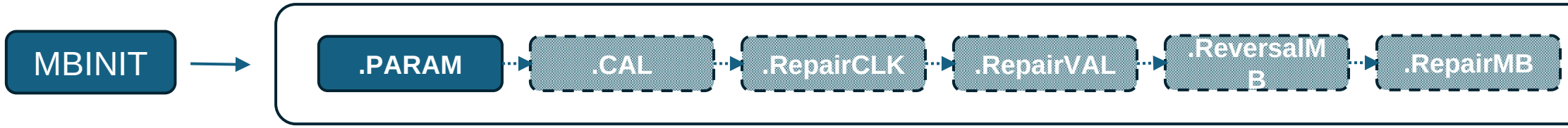

Die A (Initiator)
Mainband
Sideband
{MBINIT_PARAM_config_resp}
msg
Die B
Mainband
Sideband

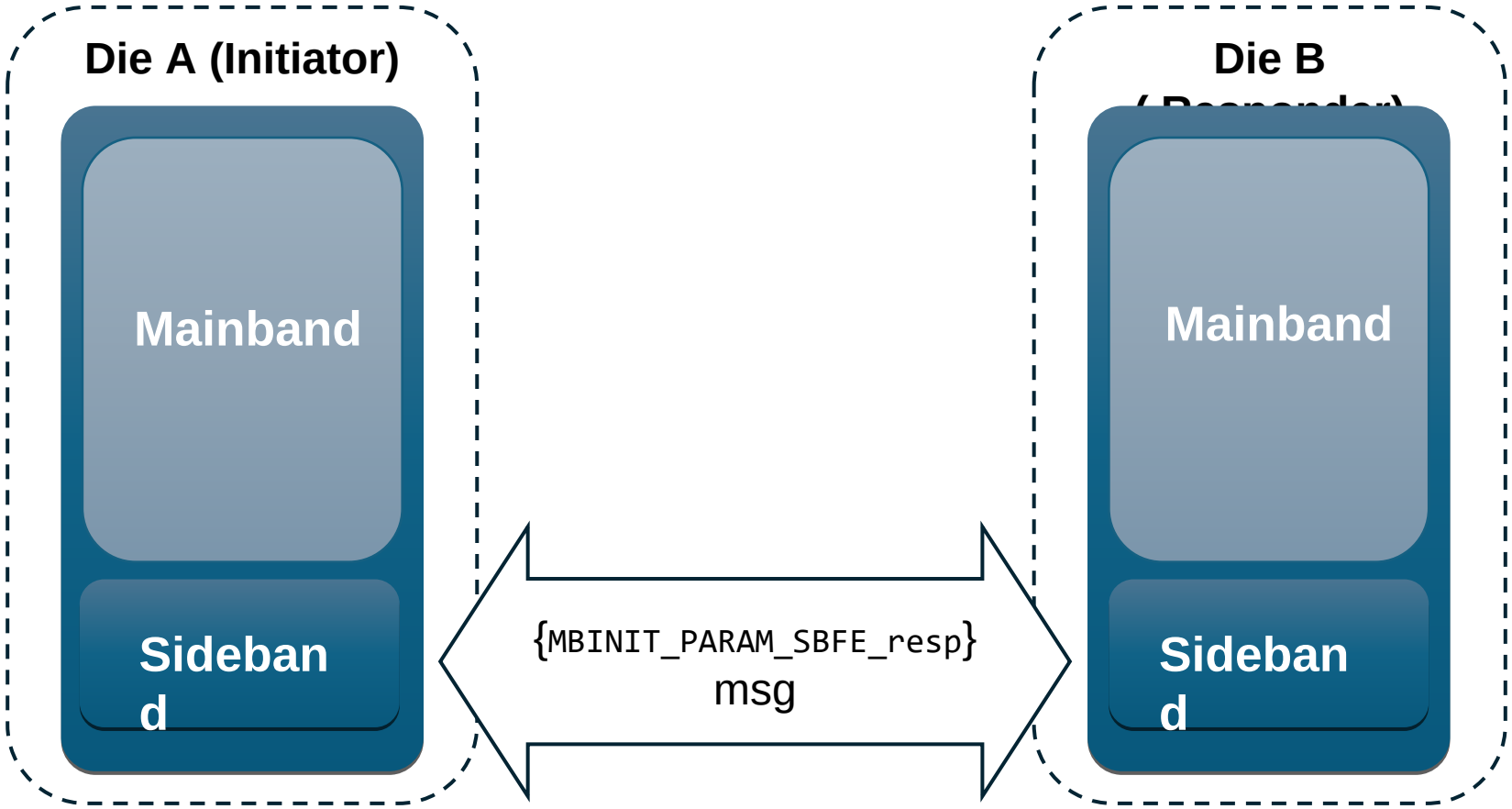

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

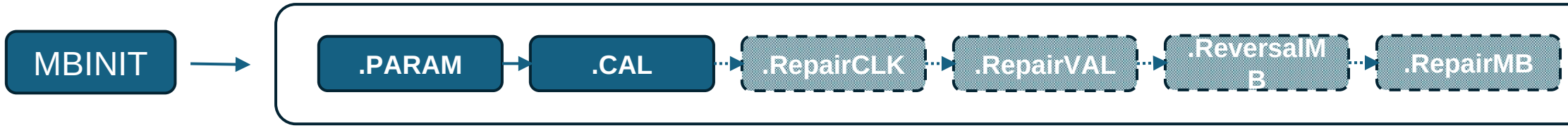

Die A (Initiator)
Mainband
Sideband
{MBINIT_PARAM_config_resp}
msg
Die B
Mainband
Sideband

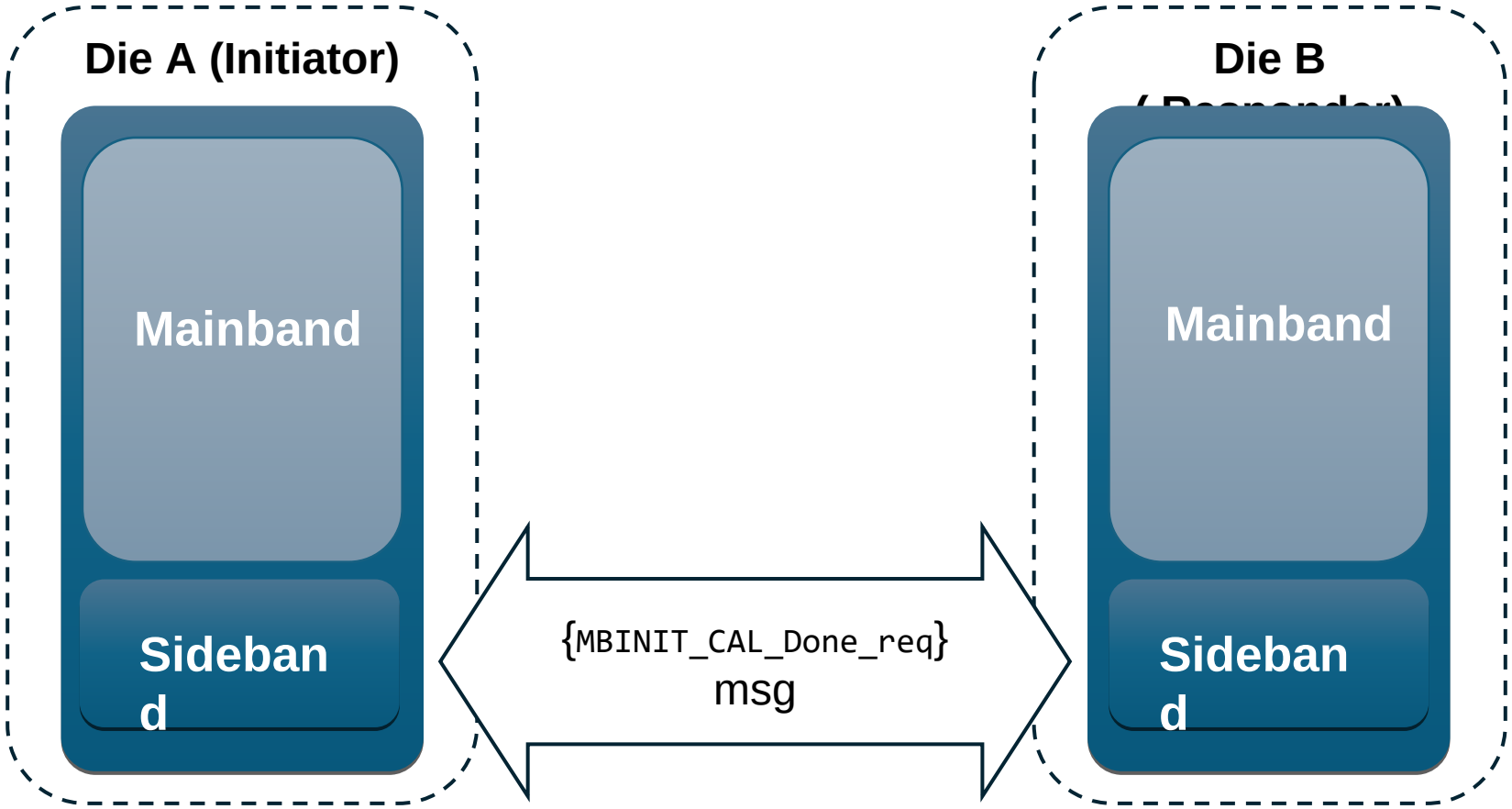

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

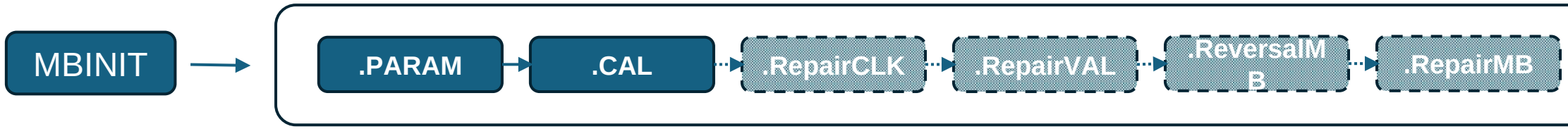

Die A (Initiator)
Mainband
Sideband
{MBINIT_PARAM_SBFE_req}
msg
Die B
Mainband
Sideband

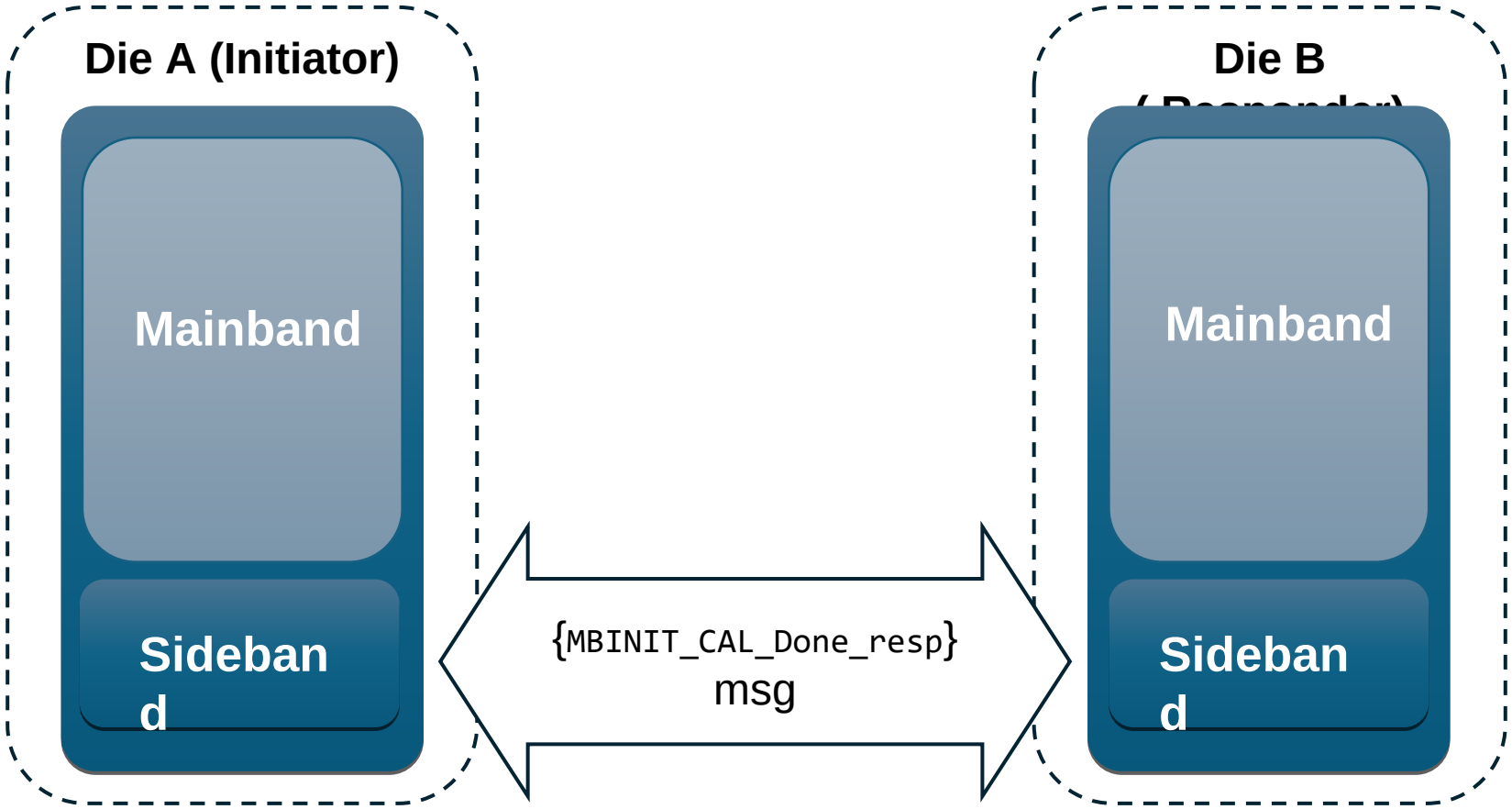

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

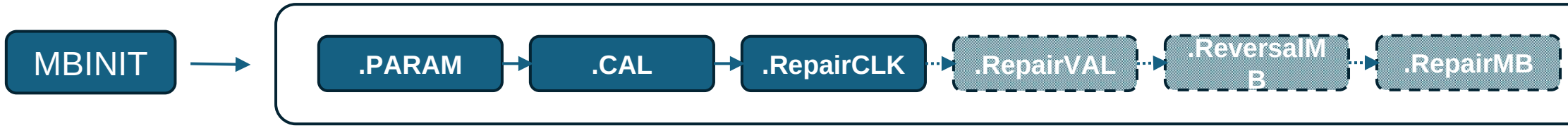

Die A (Initiator)
Mainband
Sideband
{MBINIT_PARAM_SBFE_resp}
msg
Die B
Mainband
Sideband

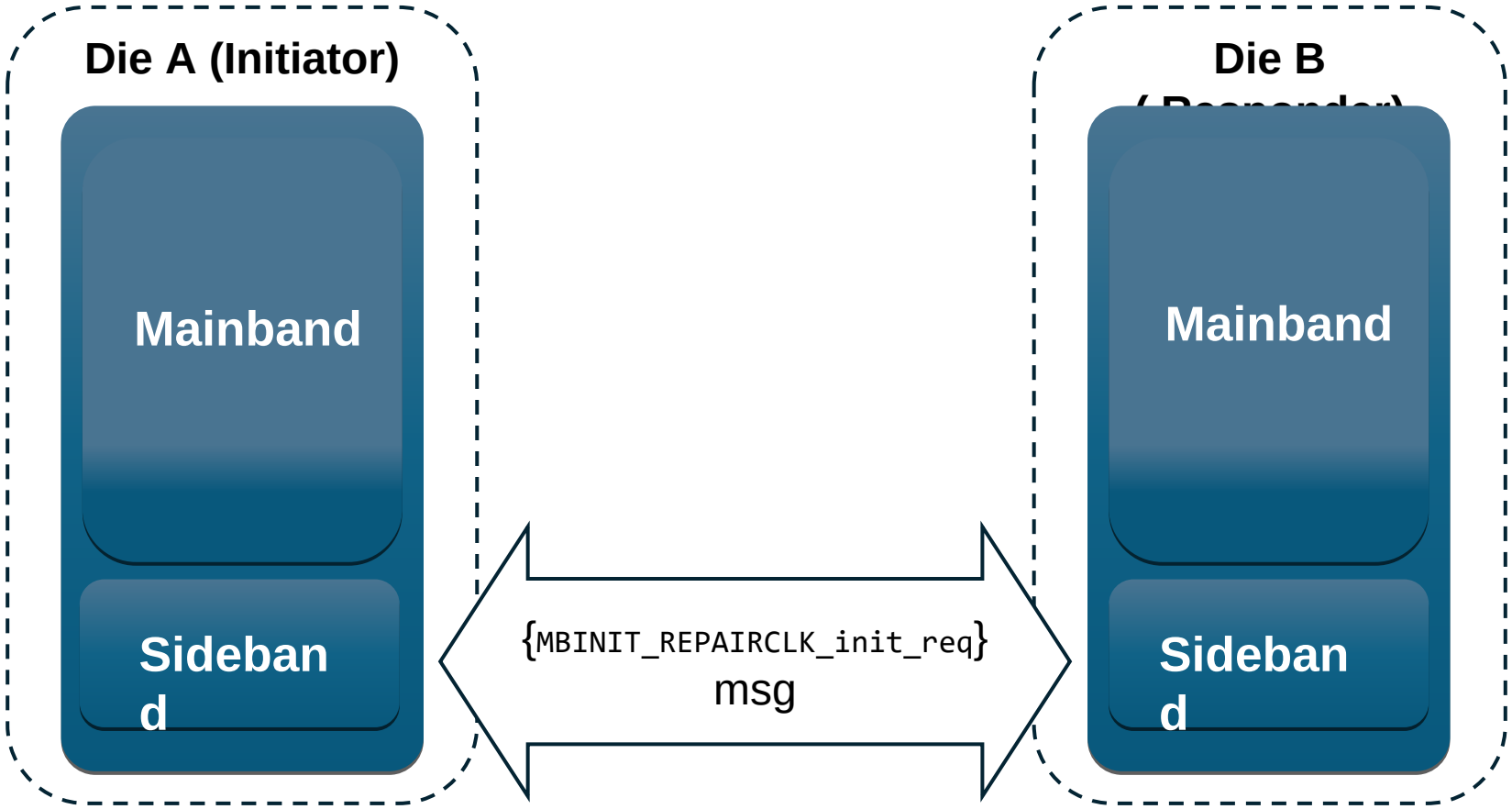

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

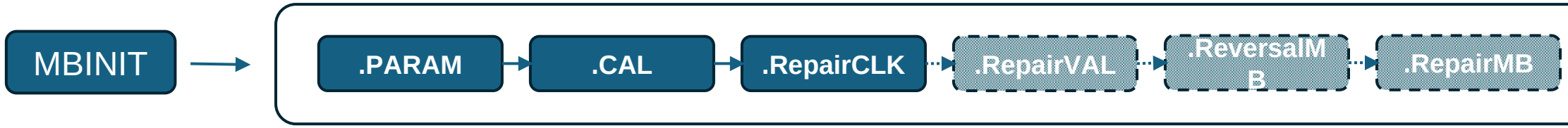

Die A (Initiator)
Mainband
Sideband
{MBINIT_CAL_Done_req}
msg
Die B
Mainband
Sideband

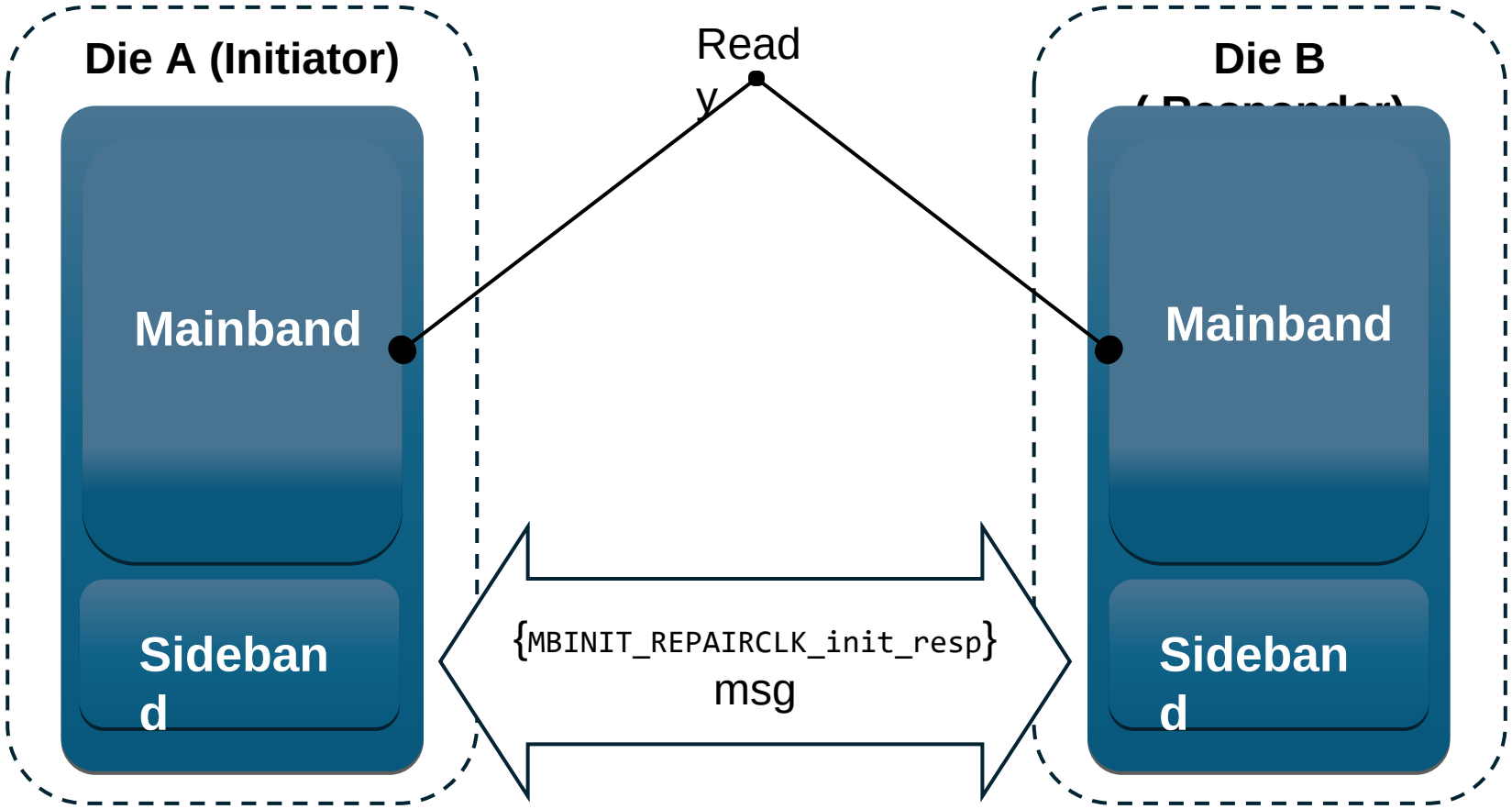

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

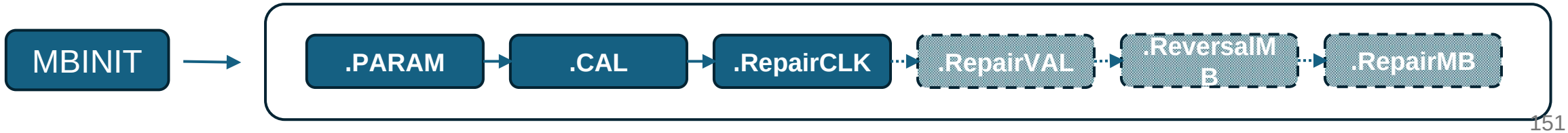

Die A (Initiator)
Mainband
Sideband
{MBINIT_CAL_Done_resp}
msg
Die B
Mainband
Sideband

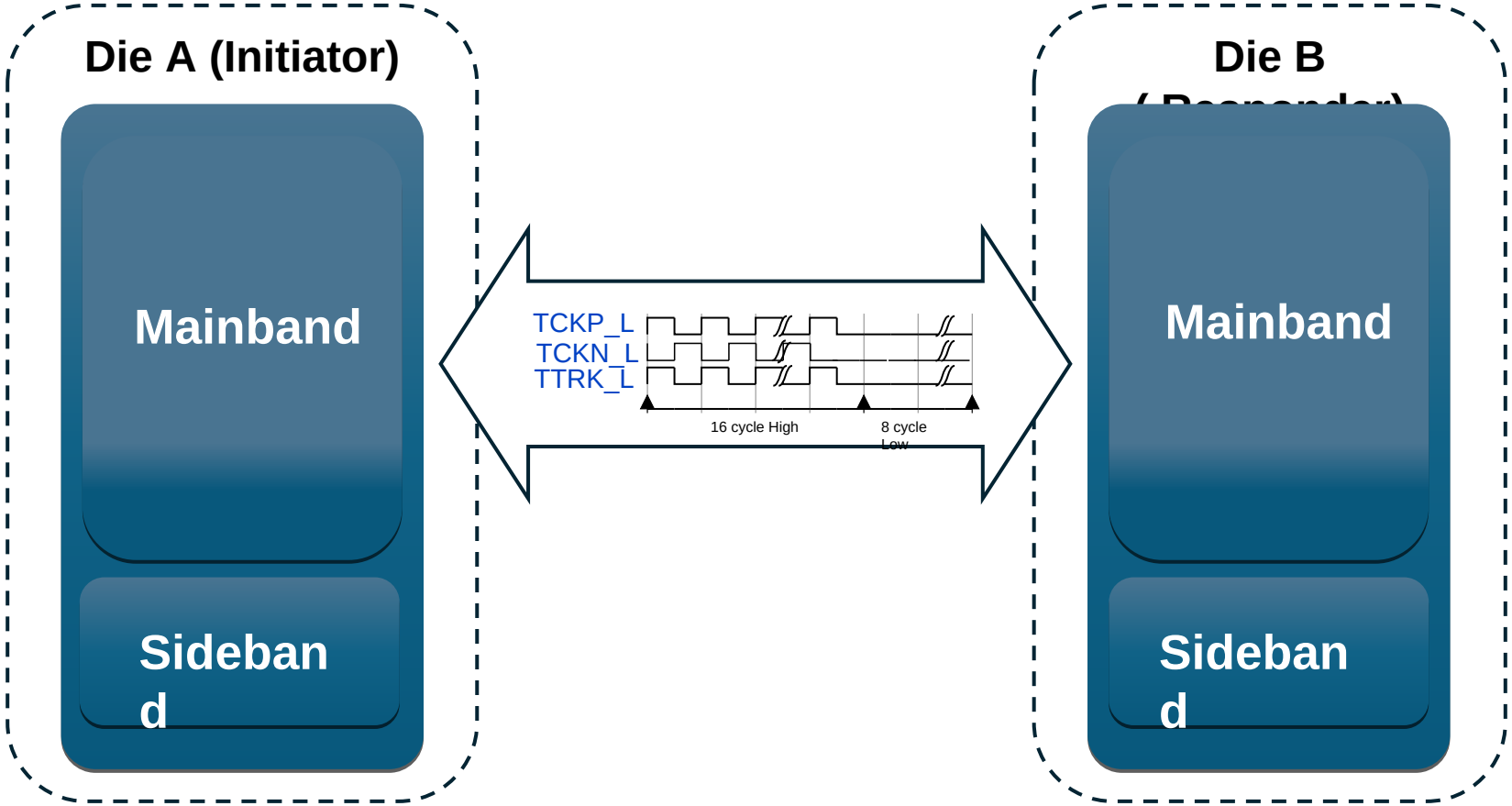

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

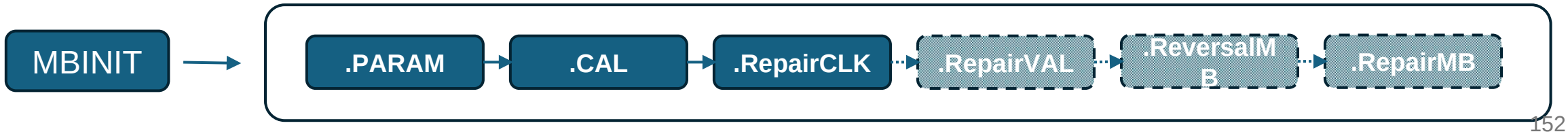

Die A (Initiator)
Mainband
Sideband
{MBINIT_REPAIRCLK_init_req}
msg
Die B
Mainband
Sideband

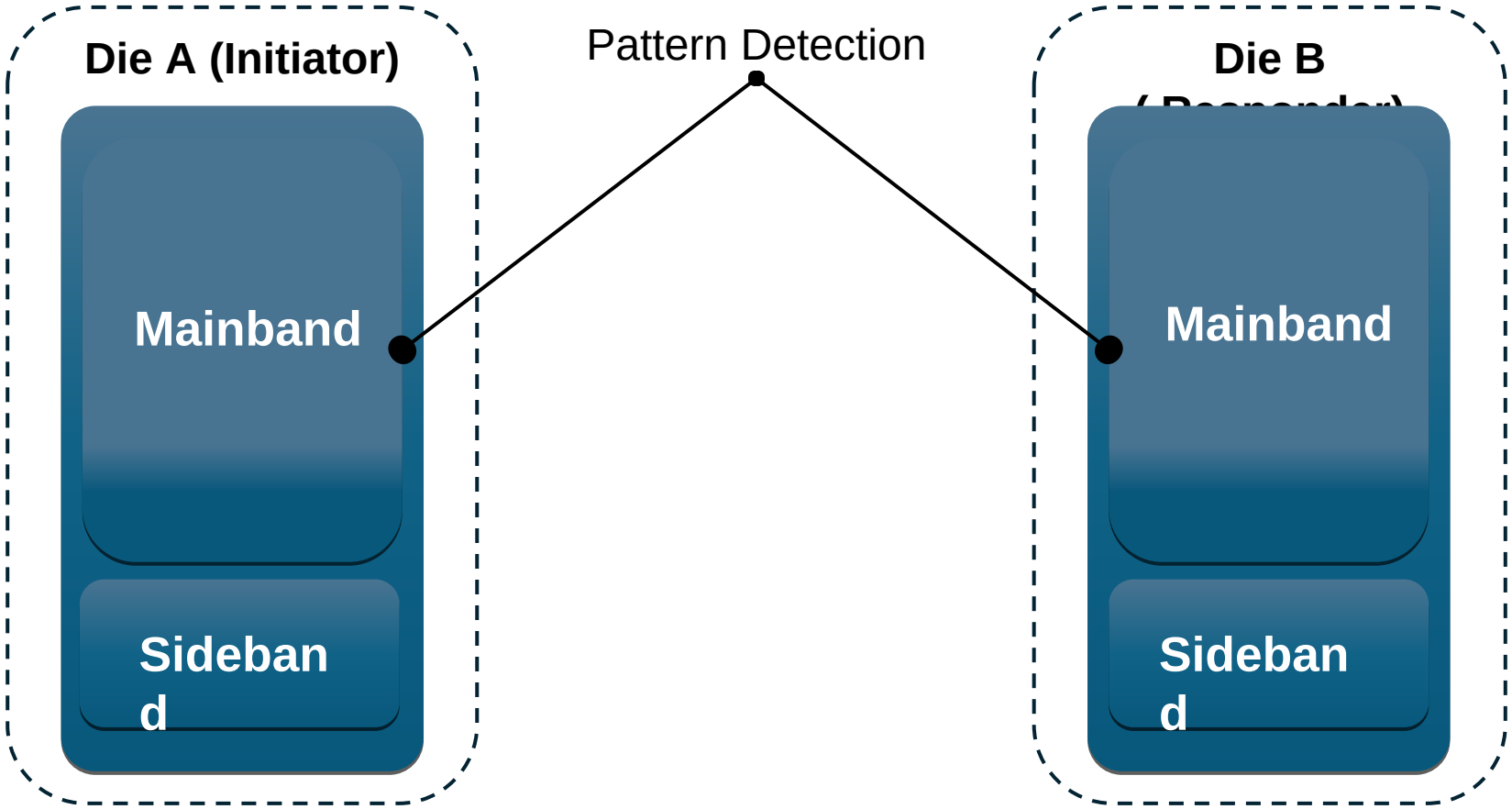

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

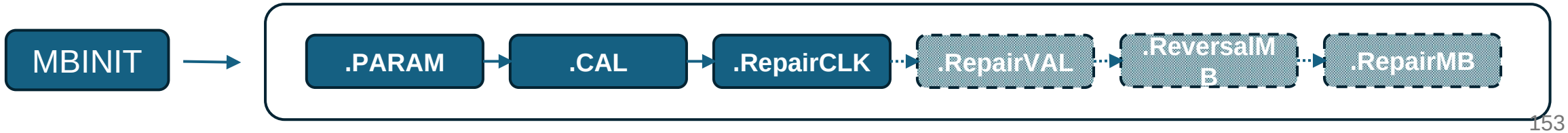

Die A (Initiator)
Ready
Die B
Mainband
Sideband
{MBINIT_REPAIRCLK_init_resp}
msg
Mainband
Sideband

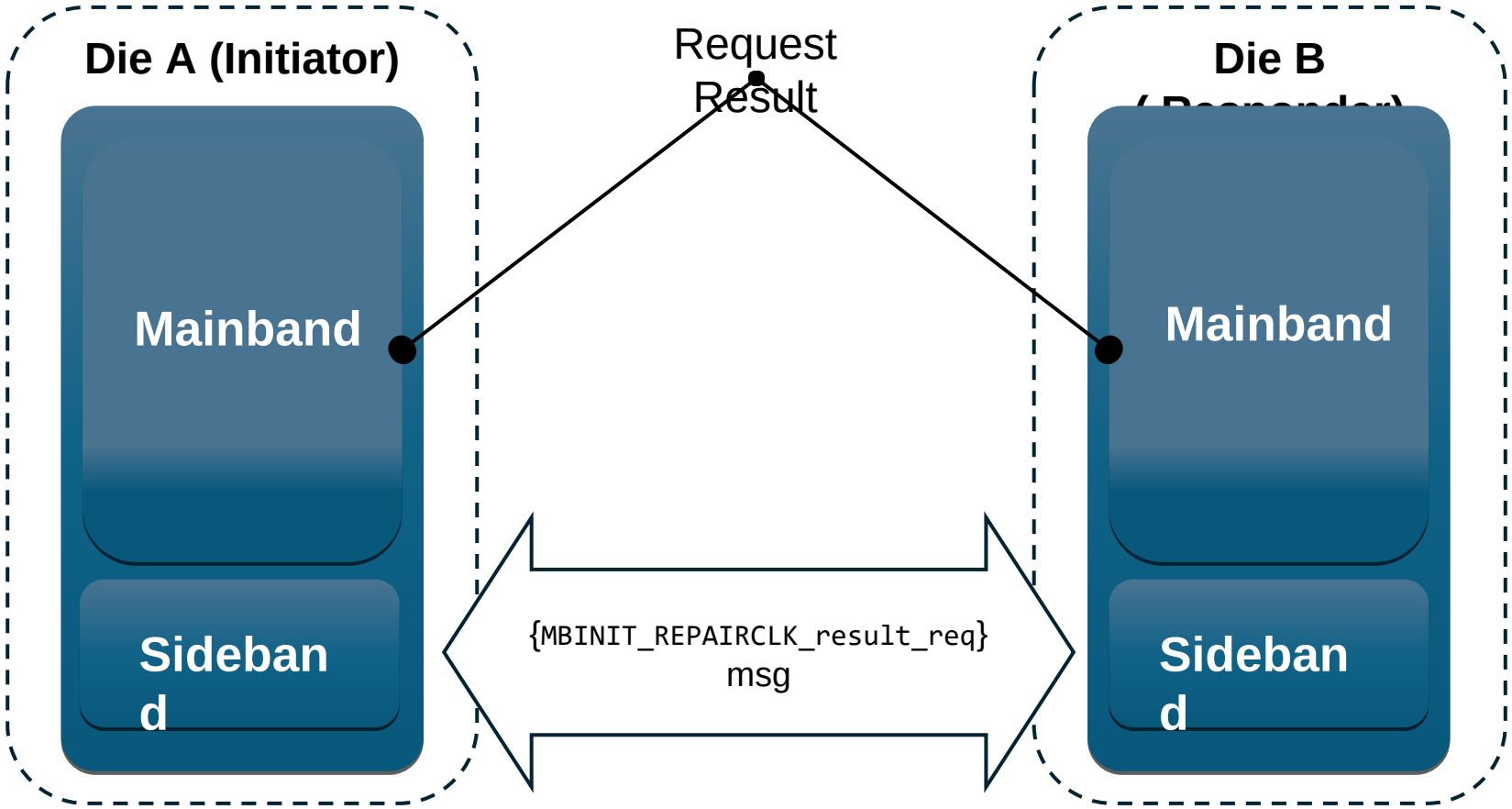

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

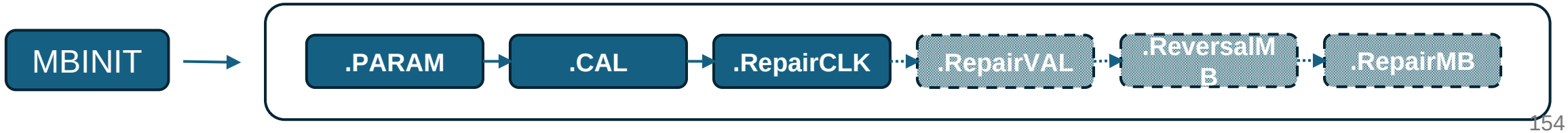

Die A (Initiator)
Mainband
Sideband
TCKP_L
TCKN_L
TTRK_L
16 cycle High
8 cycle Low
Die B
Mainband
Sideband

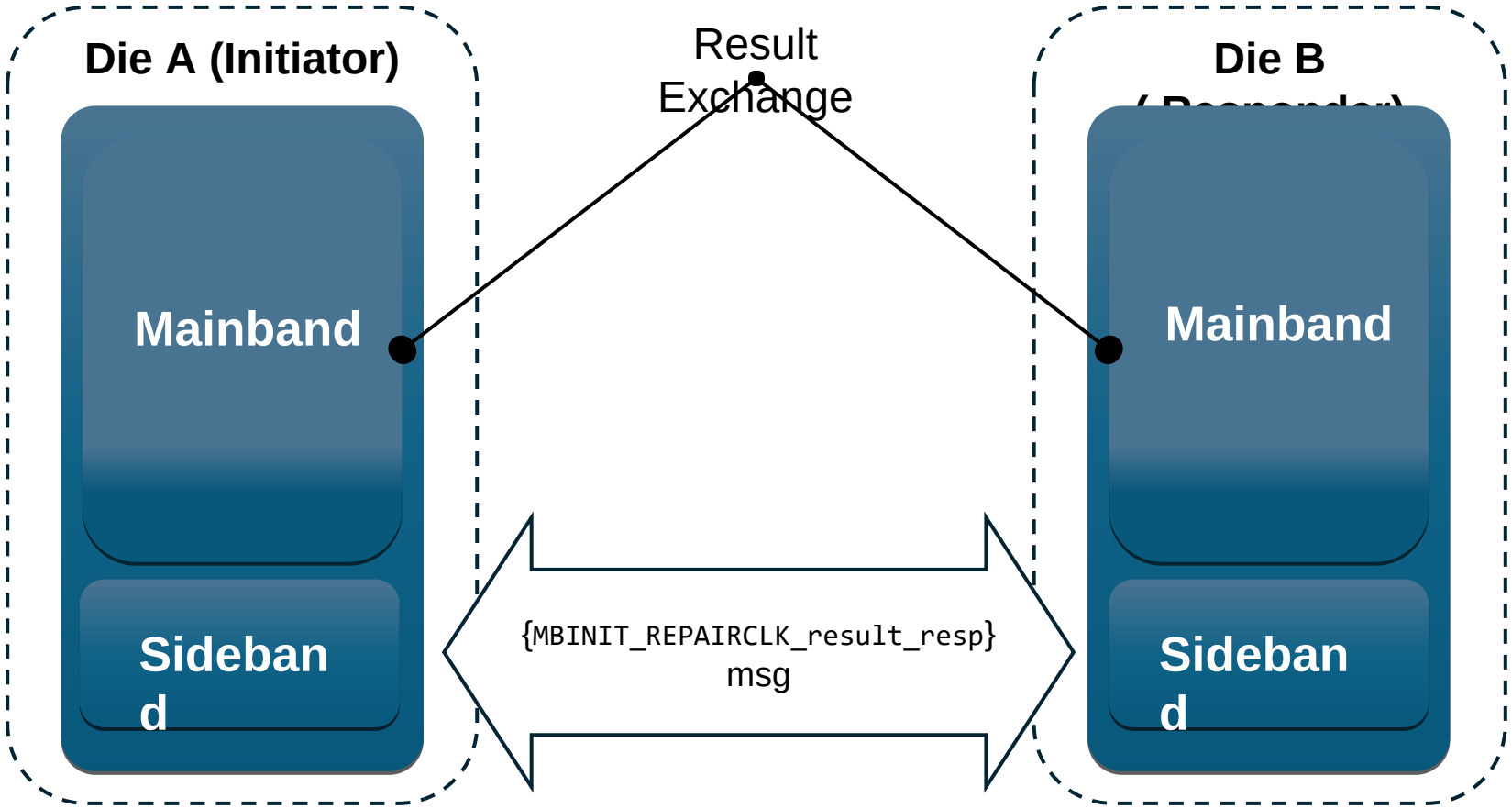

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

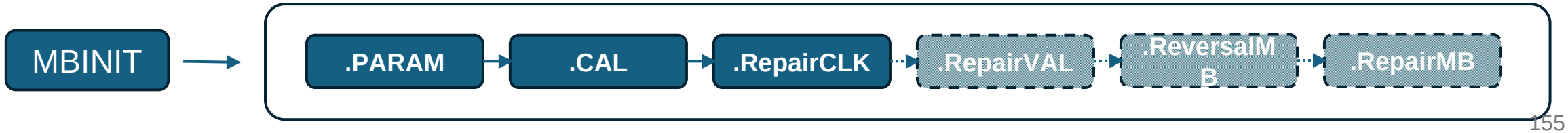

Die A (Initiator)
Pattern Detection
Die B
Mainband
Sideband
Mainband
Sideband

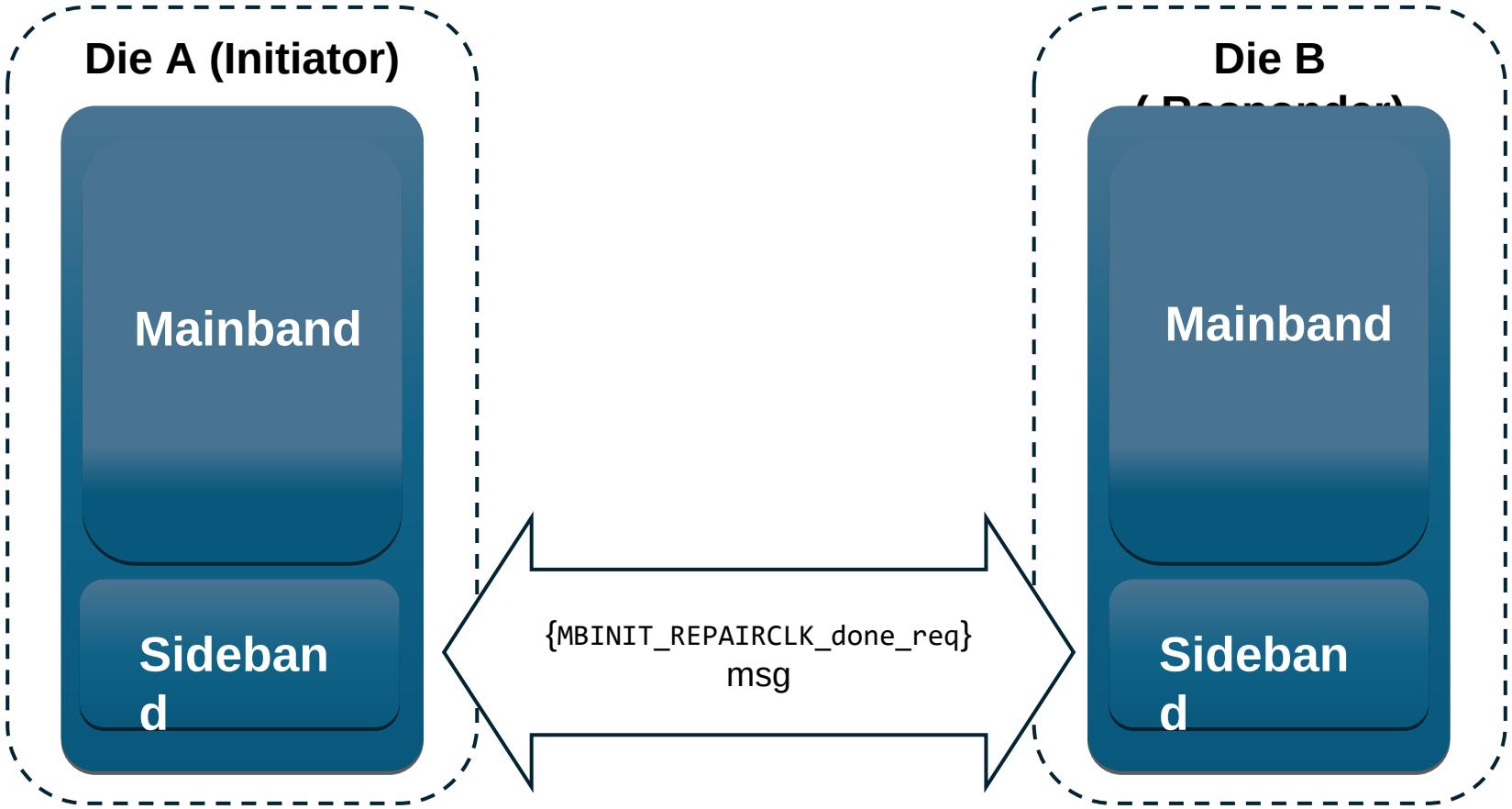

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

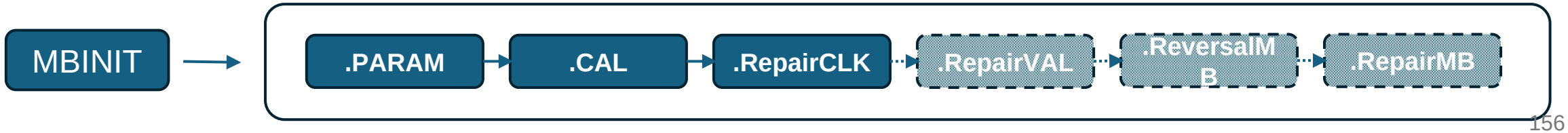

Die A (Initiator)
Request
Result
Die B
Mainband
Sideband
{MBINIT_REPAIRCLK_result_req}
msg
Mainband
Sideband

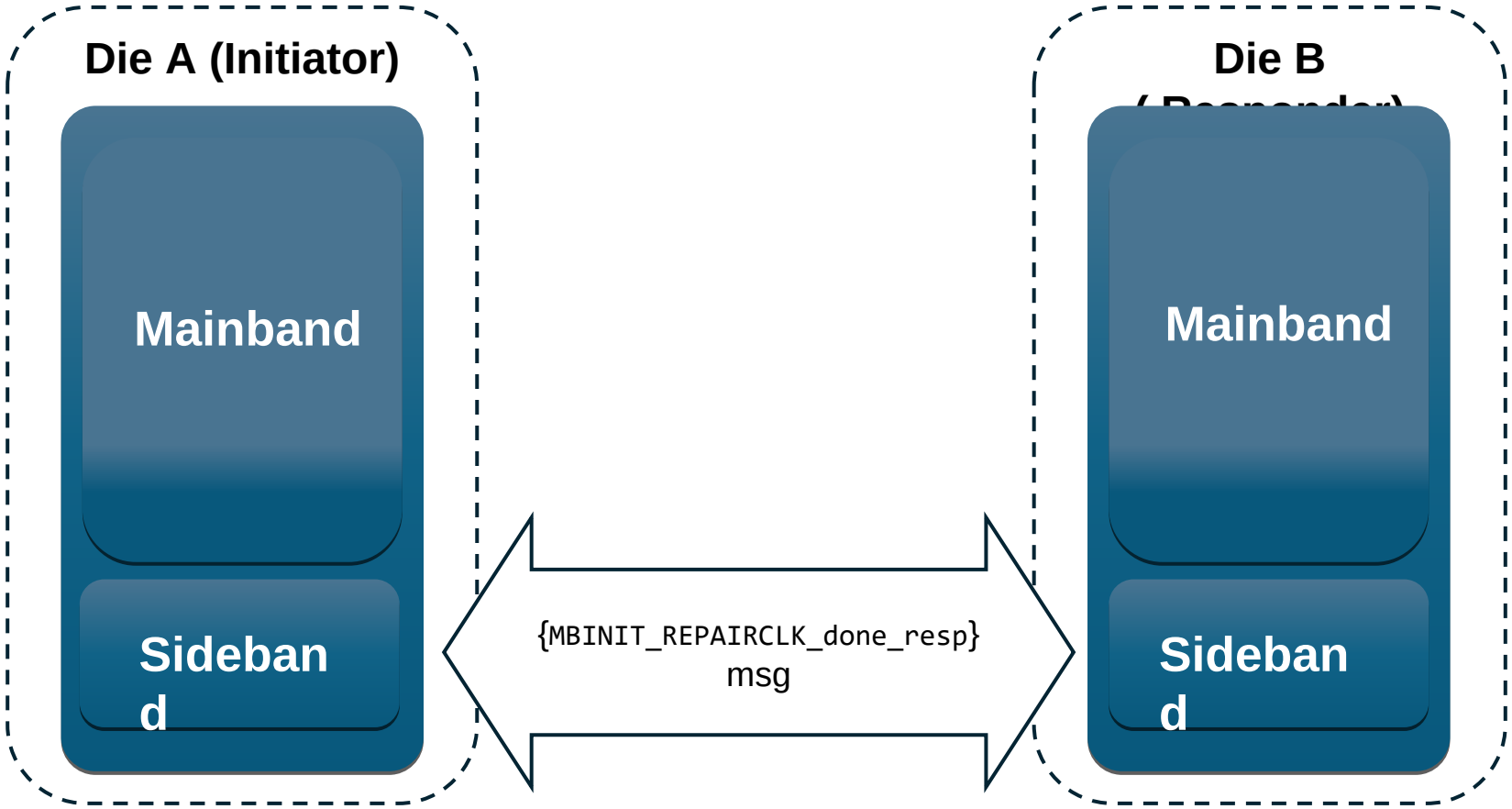

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

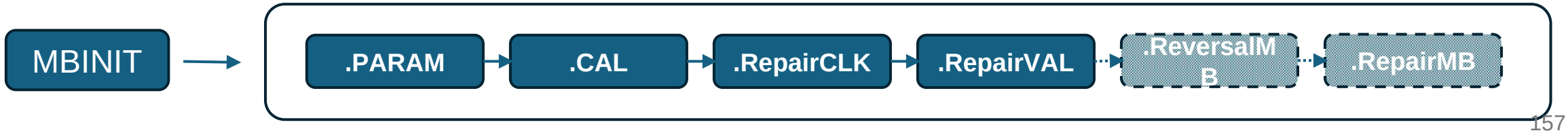

Die A (Initiator)
Result Exchange
Die B
Mainband
Sideband
Mainband
Sideband
{MBINIT_REPAIRCLK_result_resp}
msg

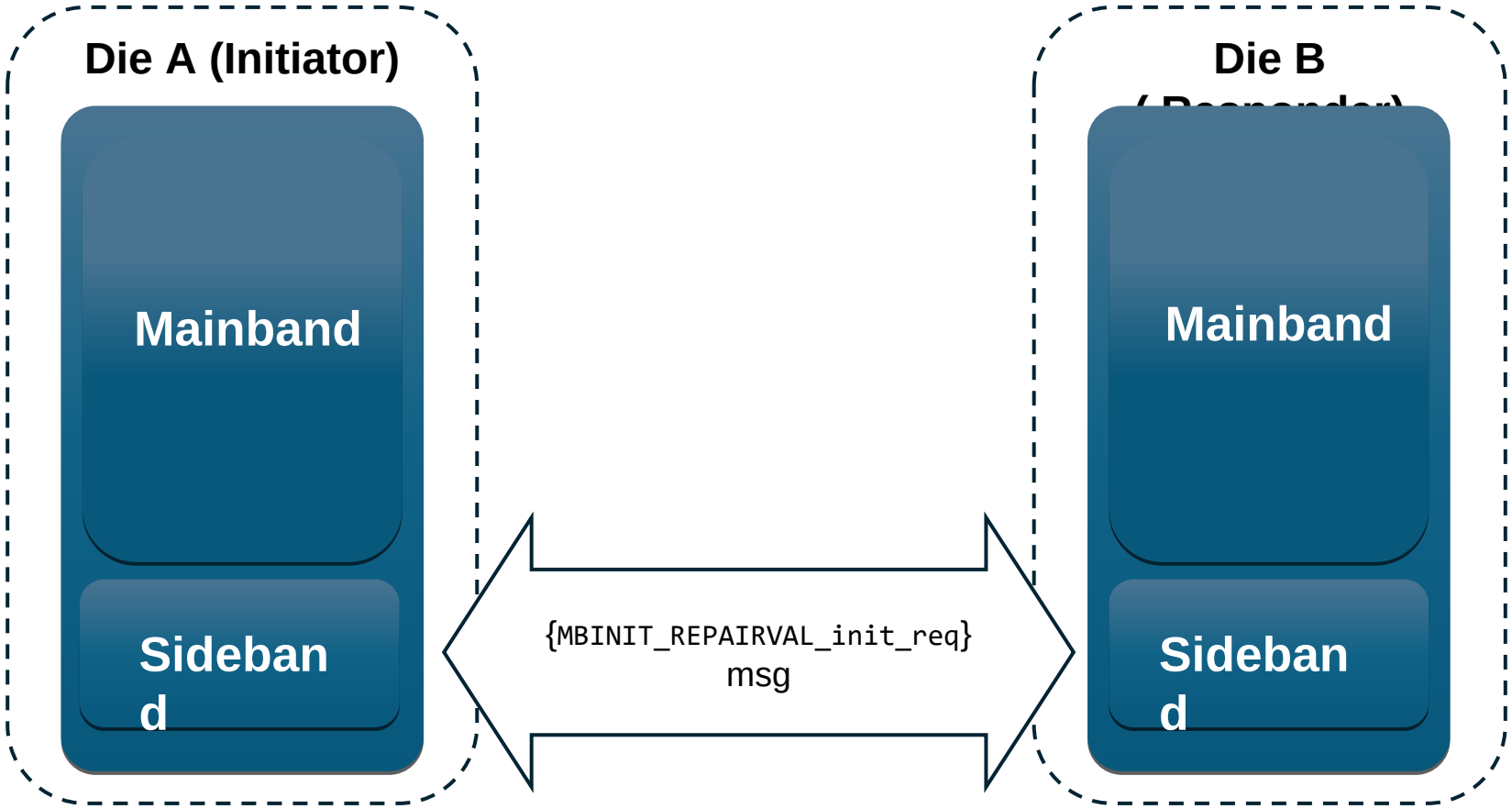

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

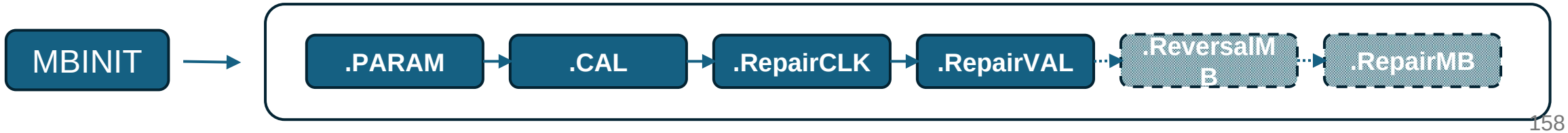

Die A (Initiator)
Mainband
Sideband
{MBINIT_REPAIRCLK_done_req}
msg
Die B
Mainband
Sideband

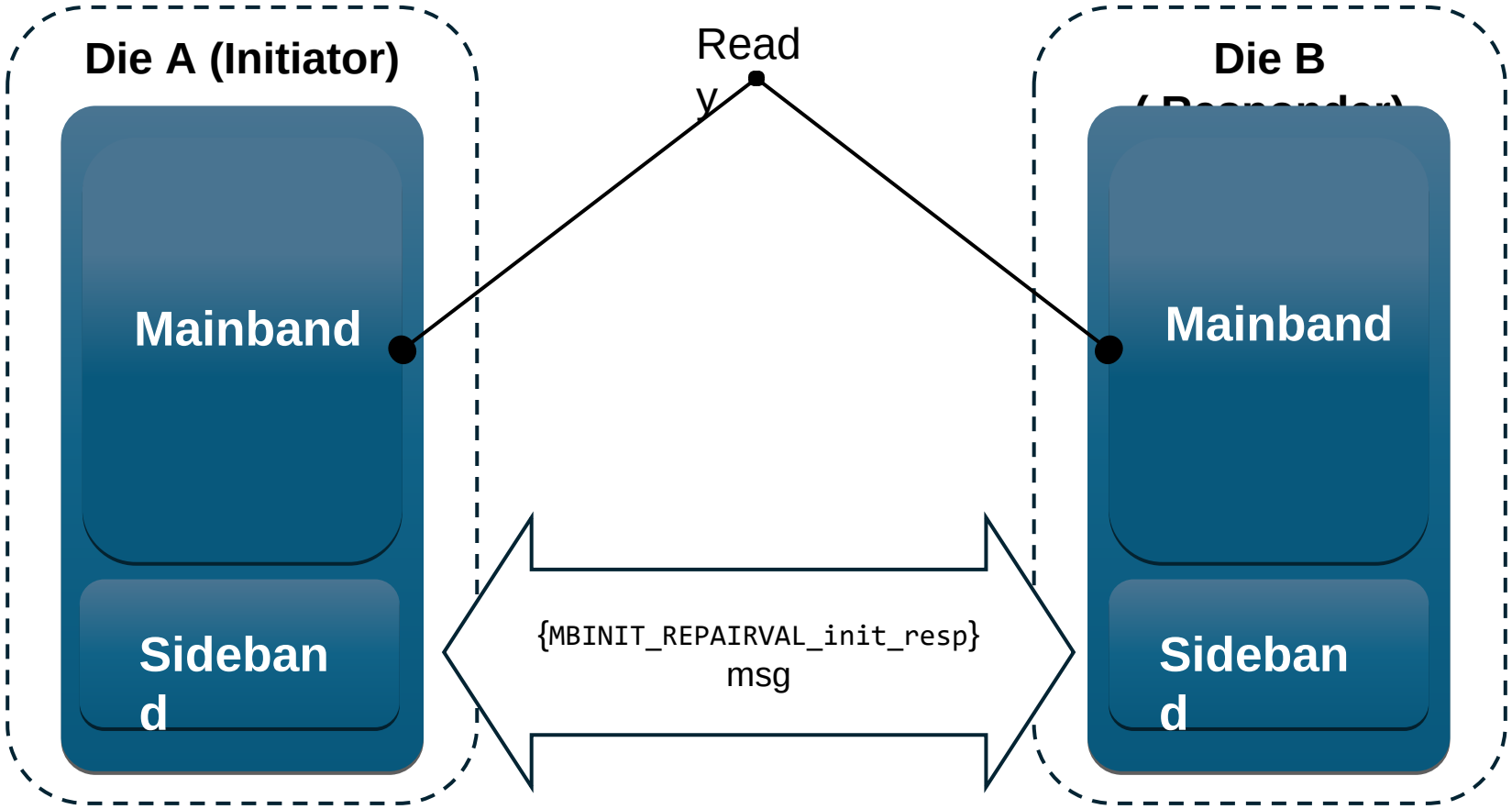

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

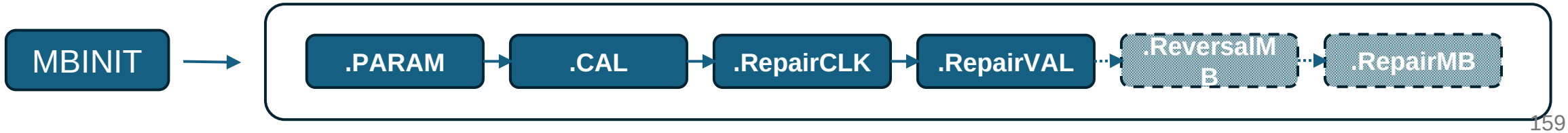

Die A (Initiator)
Mainband
Sideband
{MBINIT_REPAIRCLK_done_resp}
msg
Die B
Mainband
Sideband

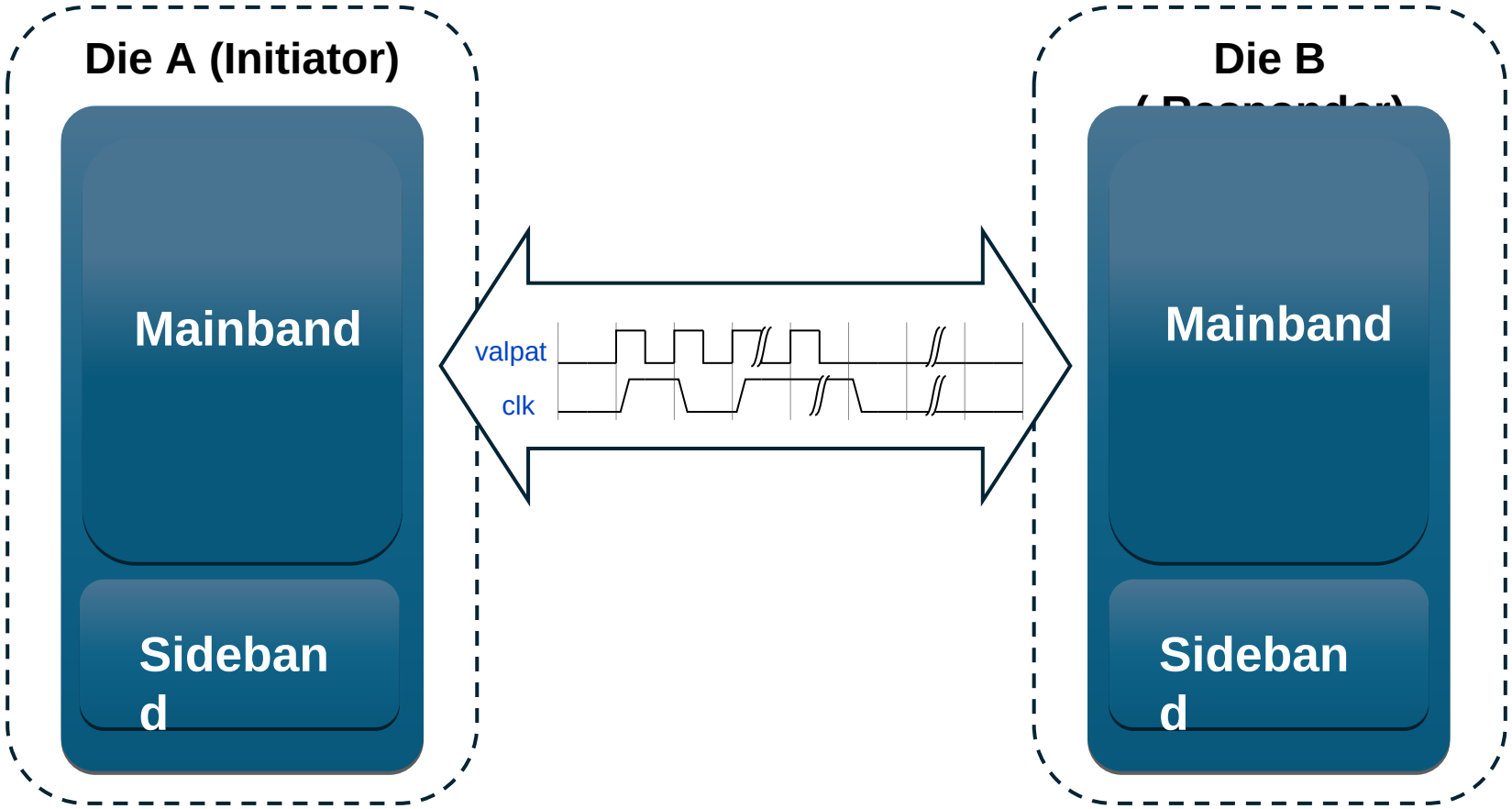

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

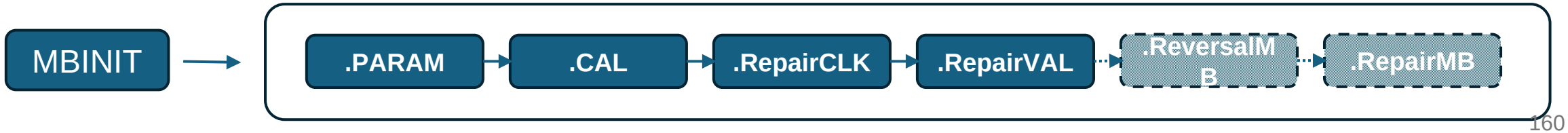

Die A (Initiator)
Mainband
Sideband
{MBINIT_REPAIRVAL_init_req}
msg
Die B
Mainband
Sideband

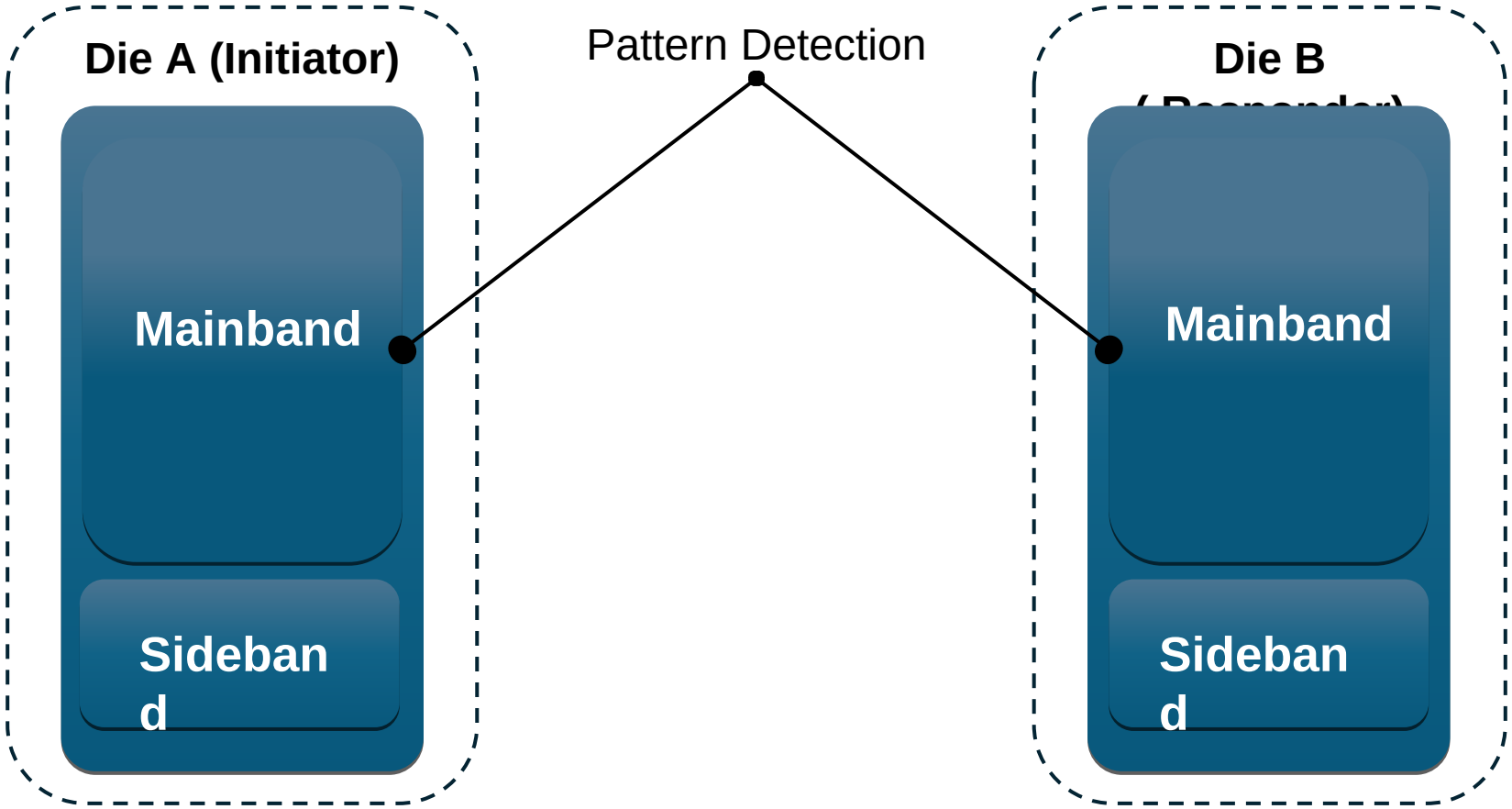

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

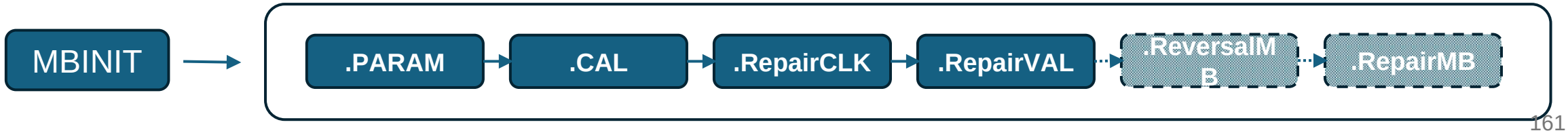

Die A (Initiator)
Ready
Die B
Mainband
Sideband
{MBINIT_REPAIRVAL_init_resp}
msg
Mainband
Sideband

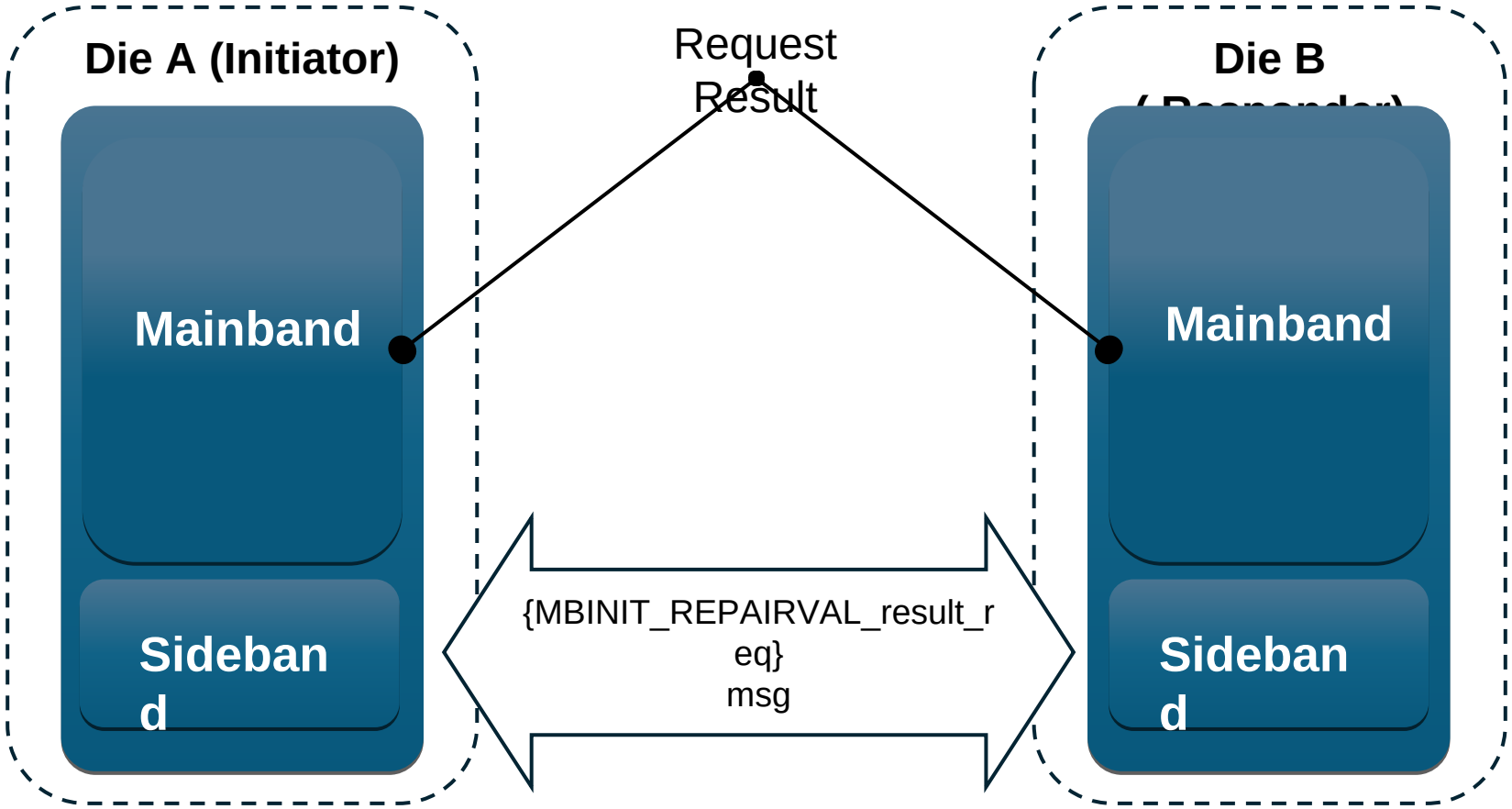

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

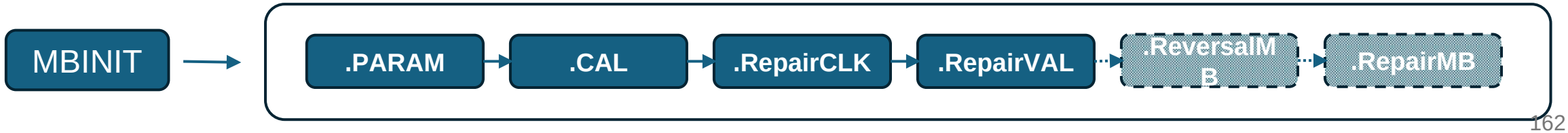

Die A (Initiator)
Mainband
Sideband
valpat
clk
Die B
Mainband
Sideband

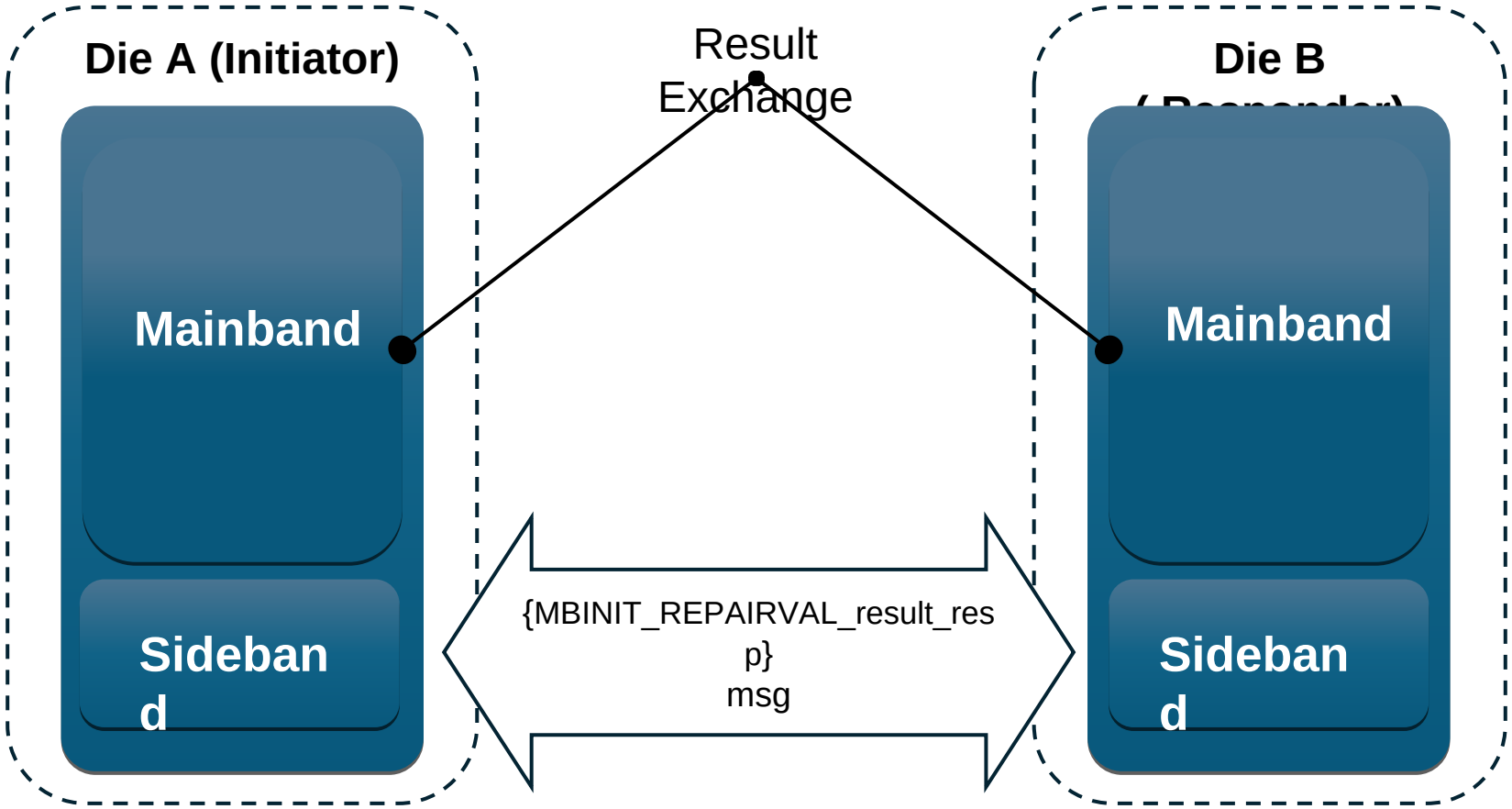

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

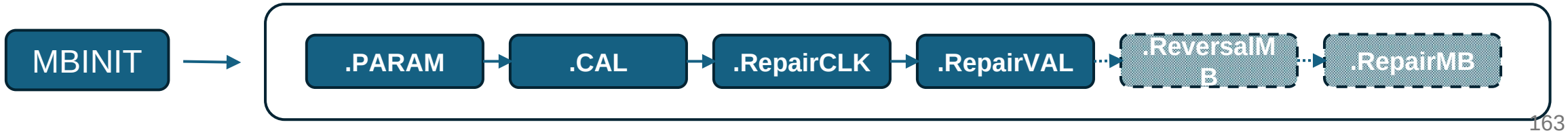

Die A (Initiator)
Pattern Detection
Die B
Mainband
Sideband
Mainband
Sideband

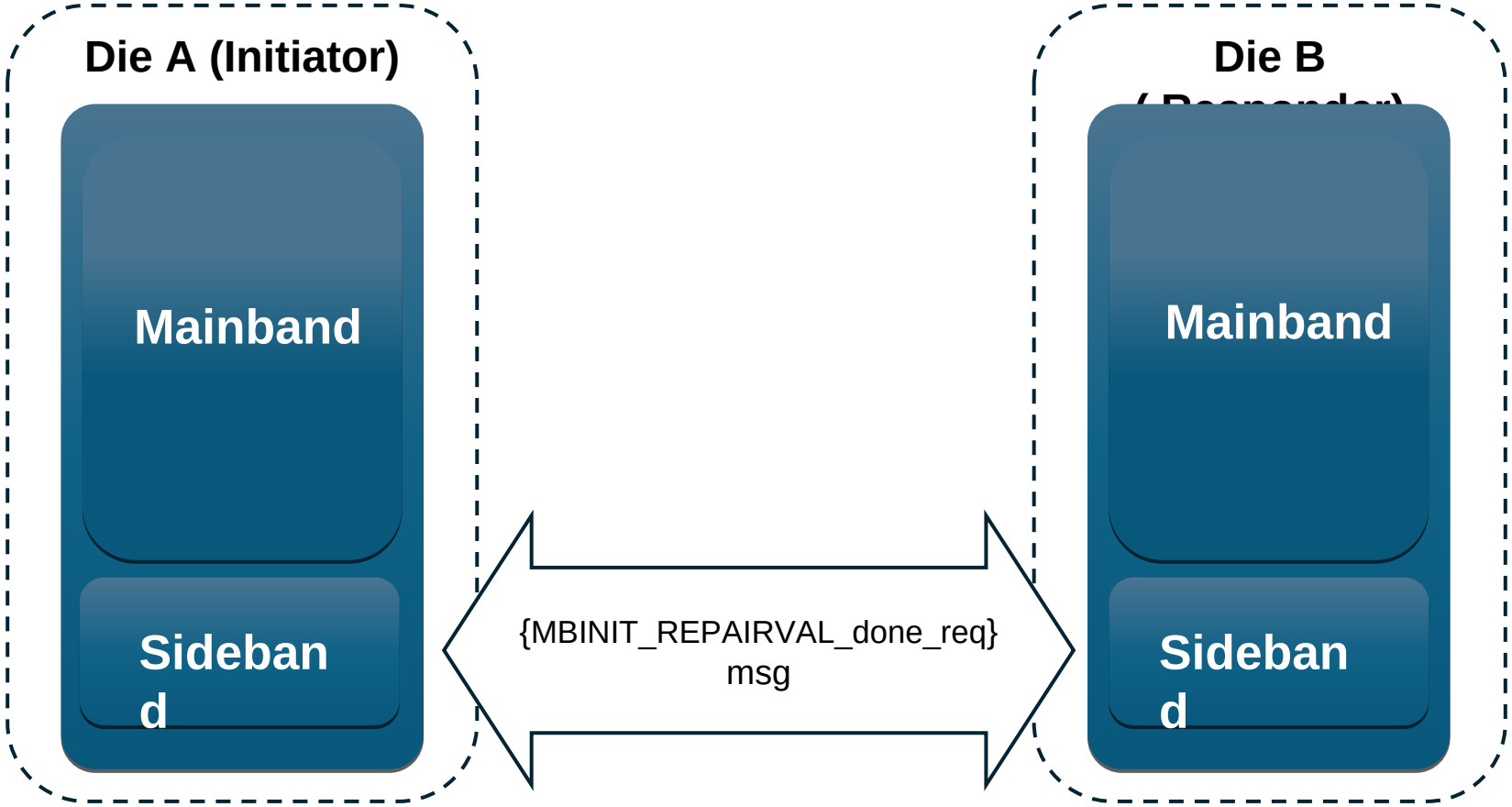

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

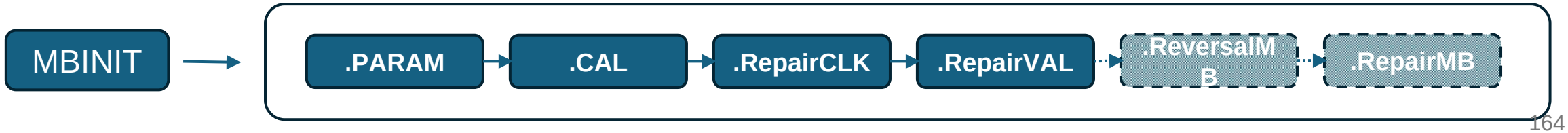

Die A (Initiator)
Request
Result
Die B
Mainband
Sideband
{MBINIT_REPAIRVAL_result_req}
msg
Mainband
Sideband

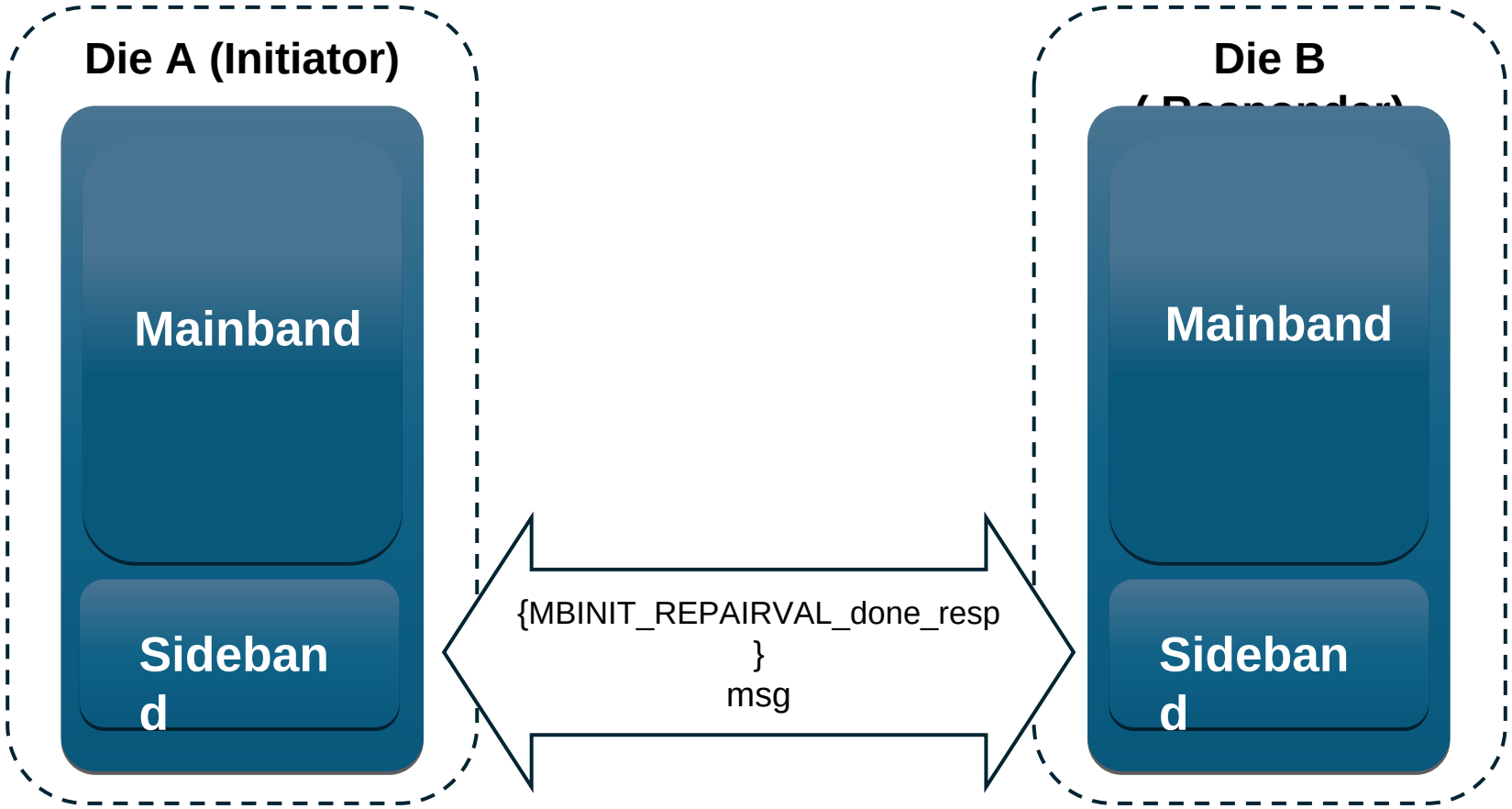

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

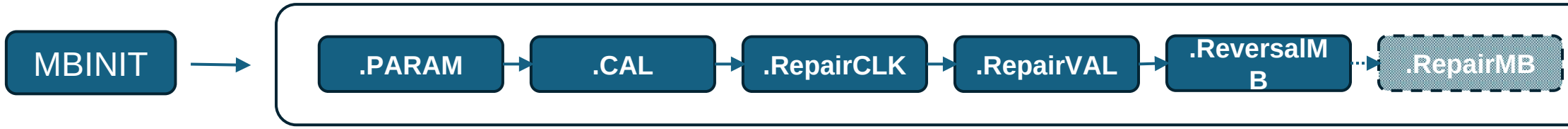

Die A (Initiator)
Result Exchange
Die B
Mainband
Sideband
{MBINIT_REPAIRVAL_result_resp}
msg
Mainband
Sideband

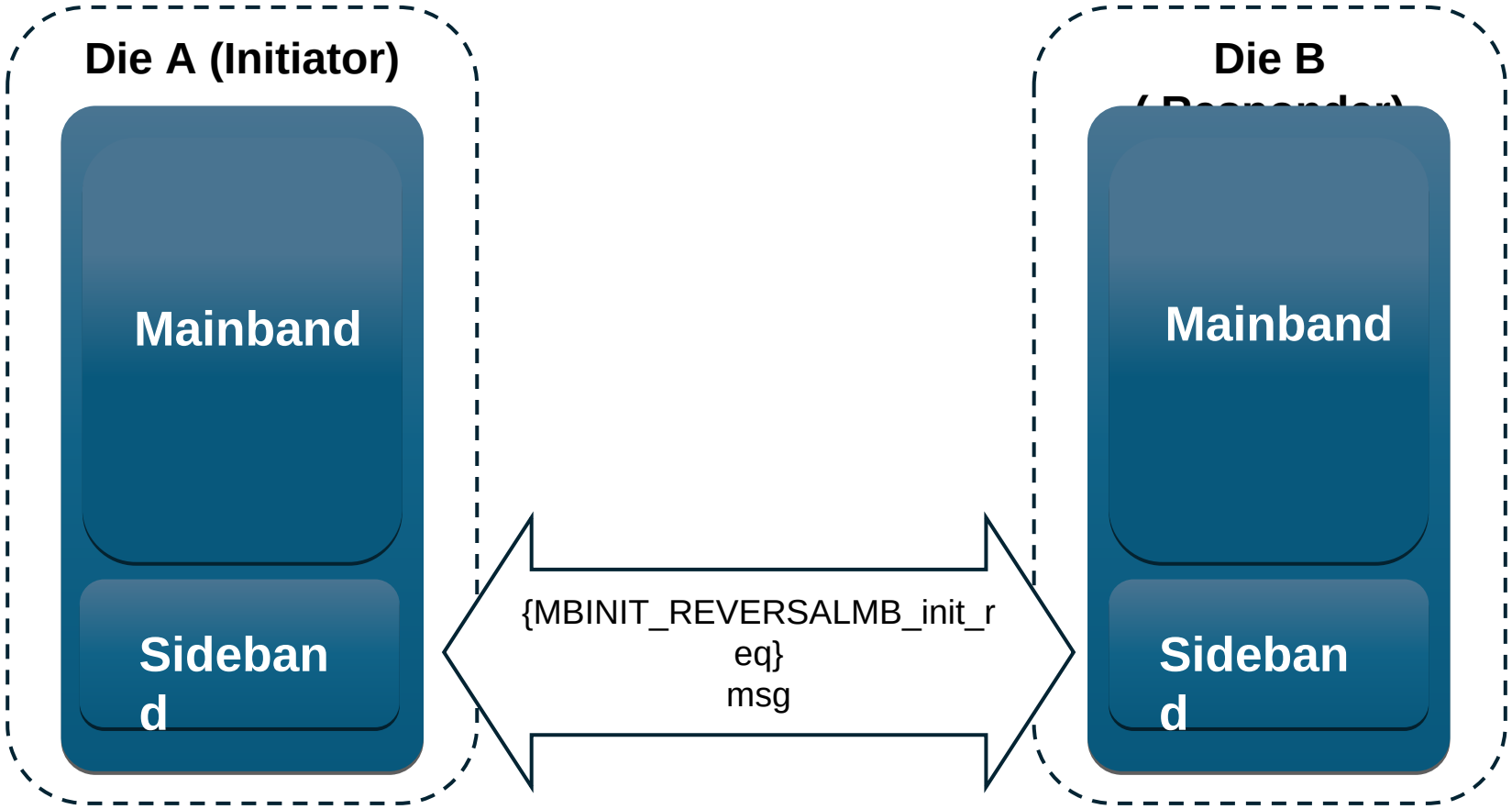

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

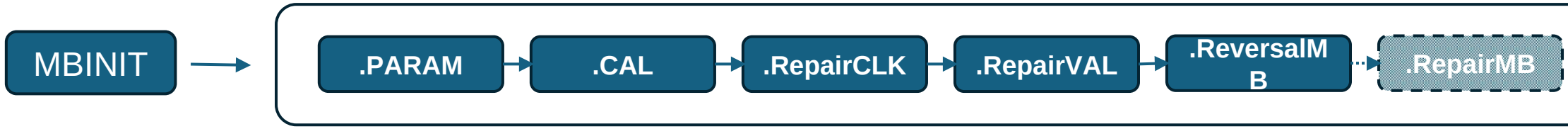

Die A (Initiator)
Mainband
Sideband
{MBINIT_REPAIRVAL_done_req}
msg
Die B
Mainband
Sideband

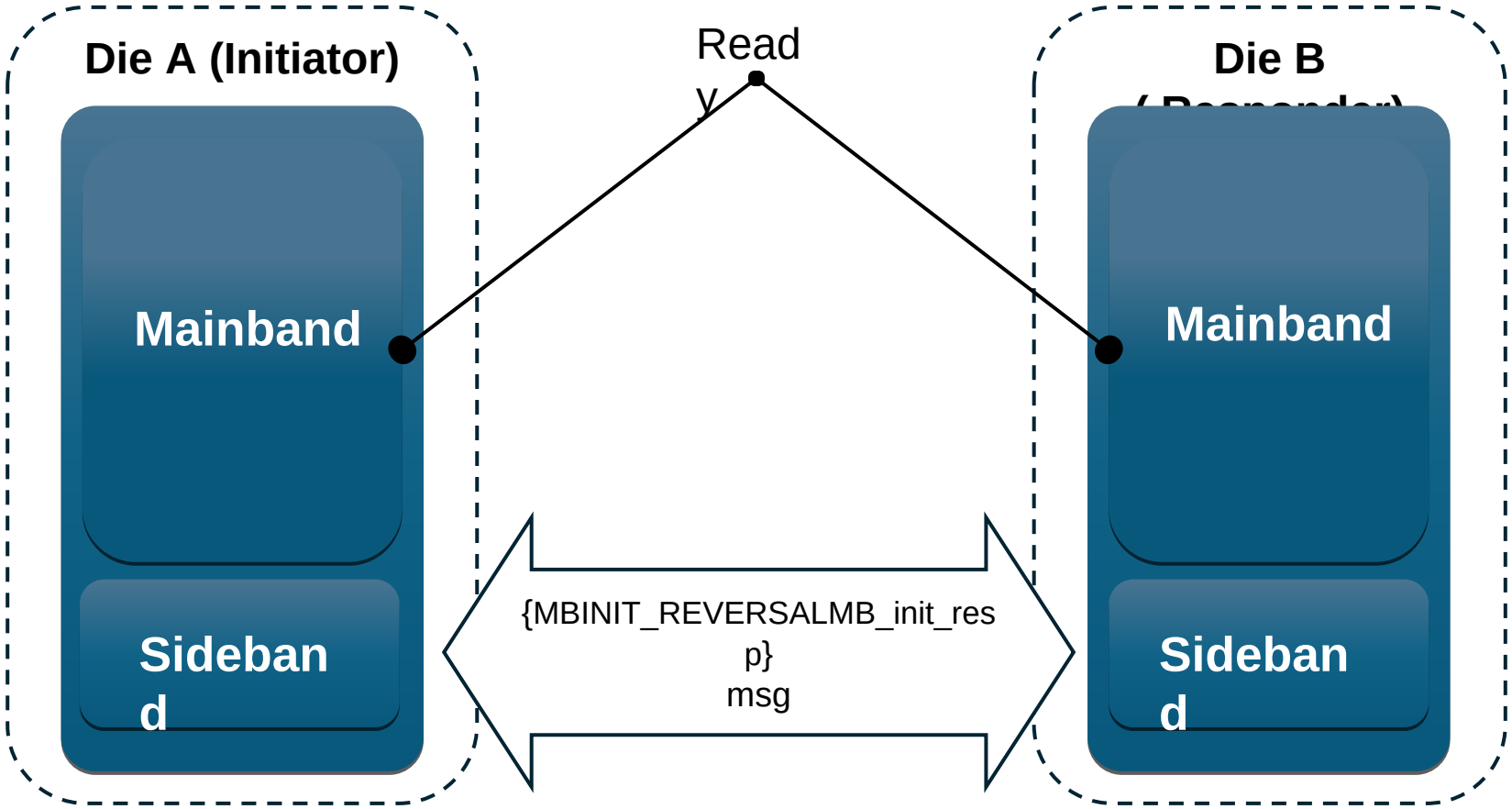

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

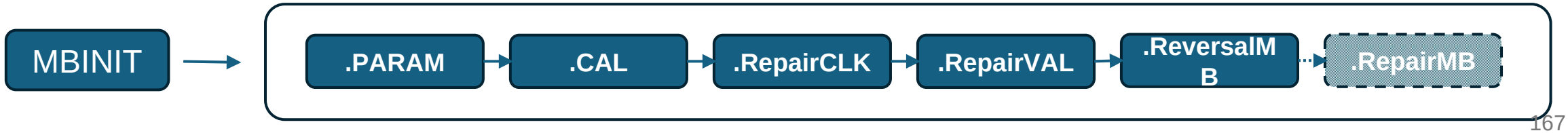

Die A (Initiator)
Mainband
Sideband
{MBINIT_REPAIRVAL_done_resp
}
msg
Die B
Mainband
Sideband

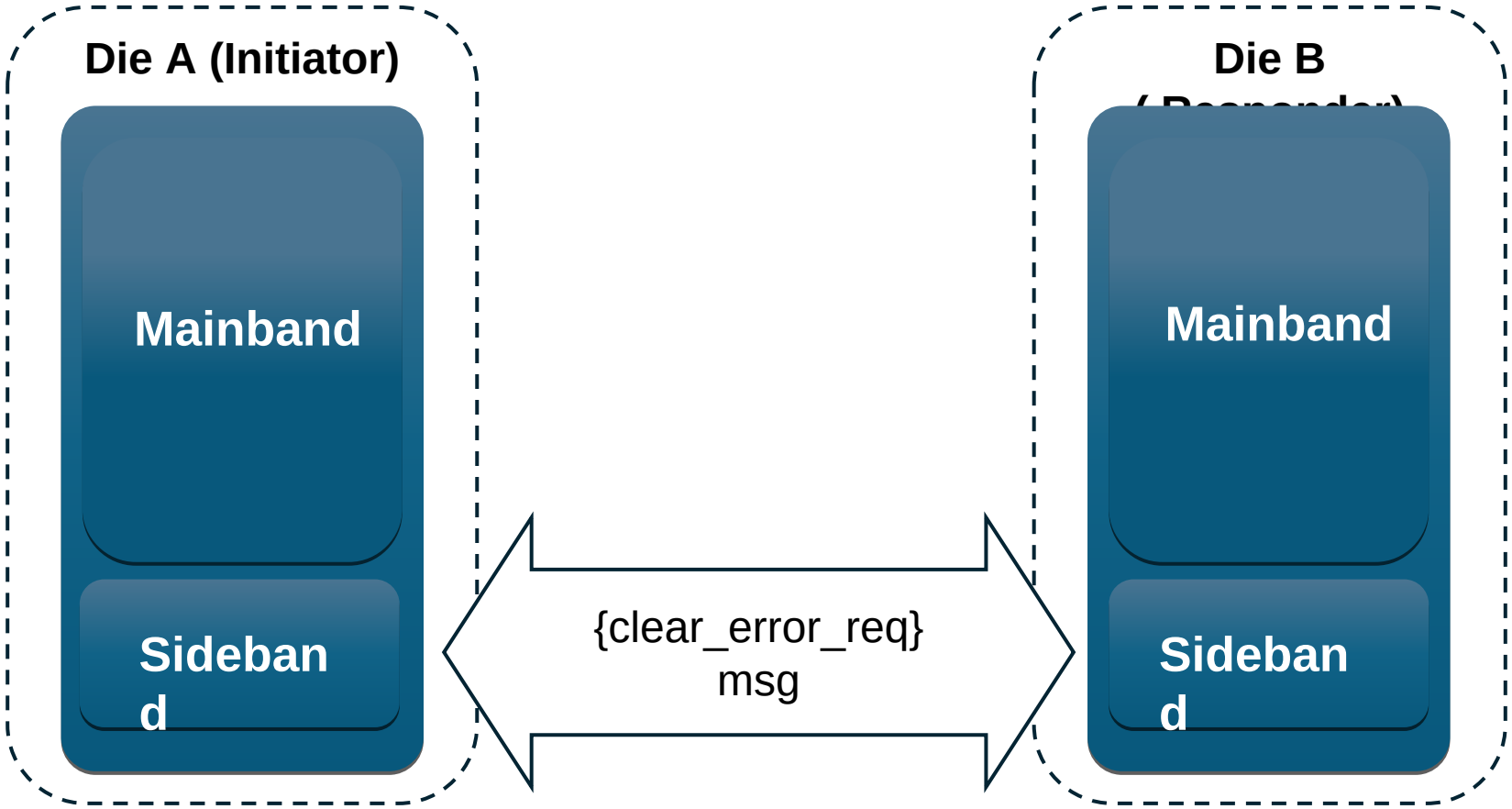

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

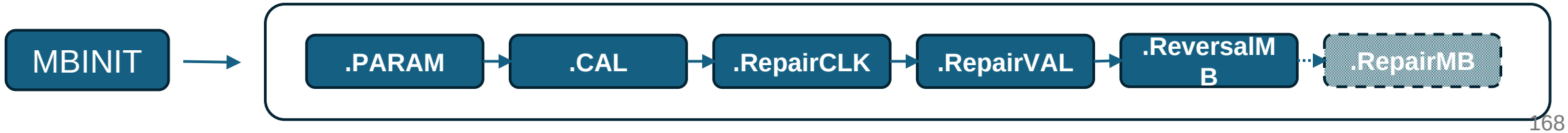

Die A (Initiator)
Mainband
Sideband
{MBINIT_REVERSALMB_init_req}
msg
Die B
Mainband
Sideband

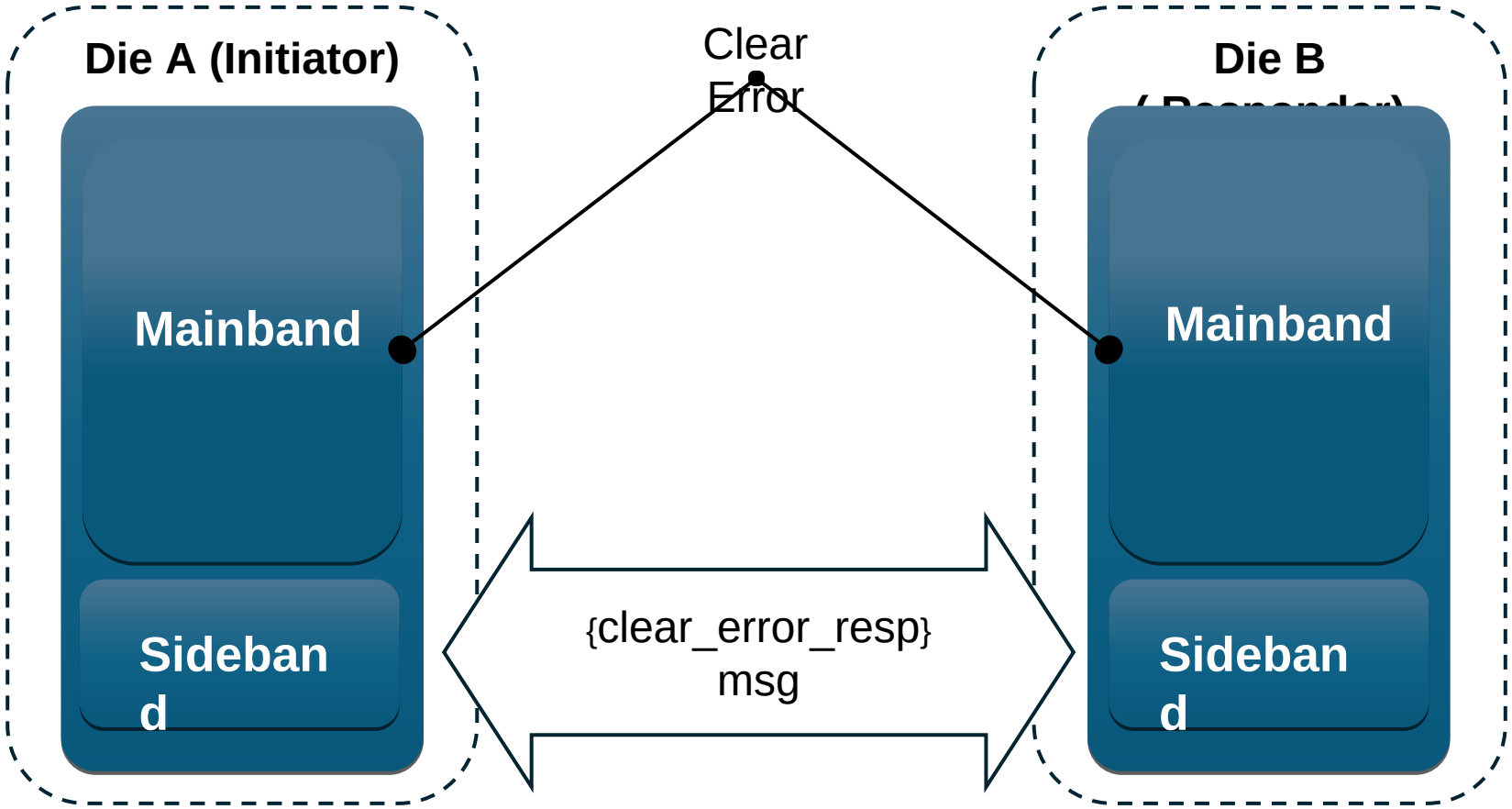

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

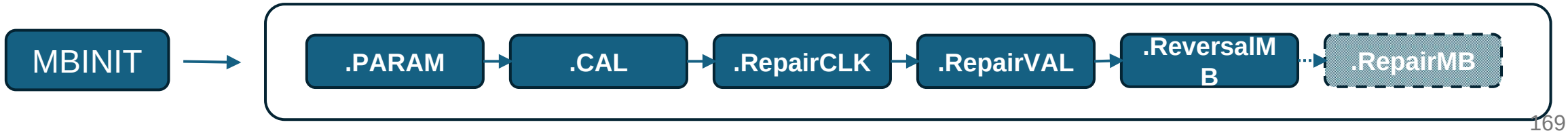

Die A (Initiator)
Ready
Die B
Mainband
Sideband
{MBINIT_REVERSALMB_init_resp}
msg
Mainband
Sideband

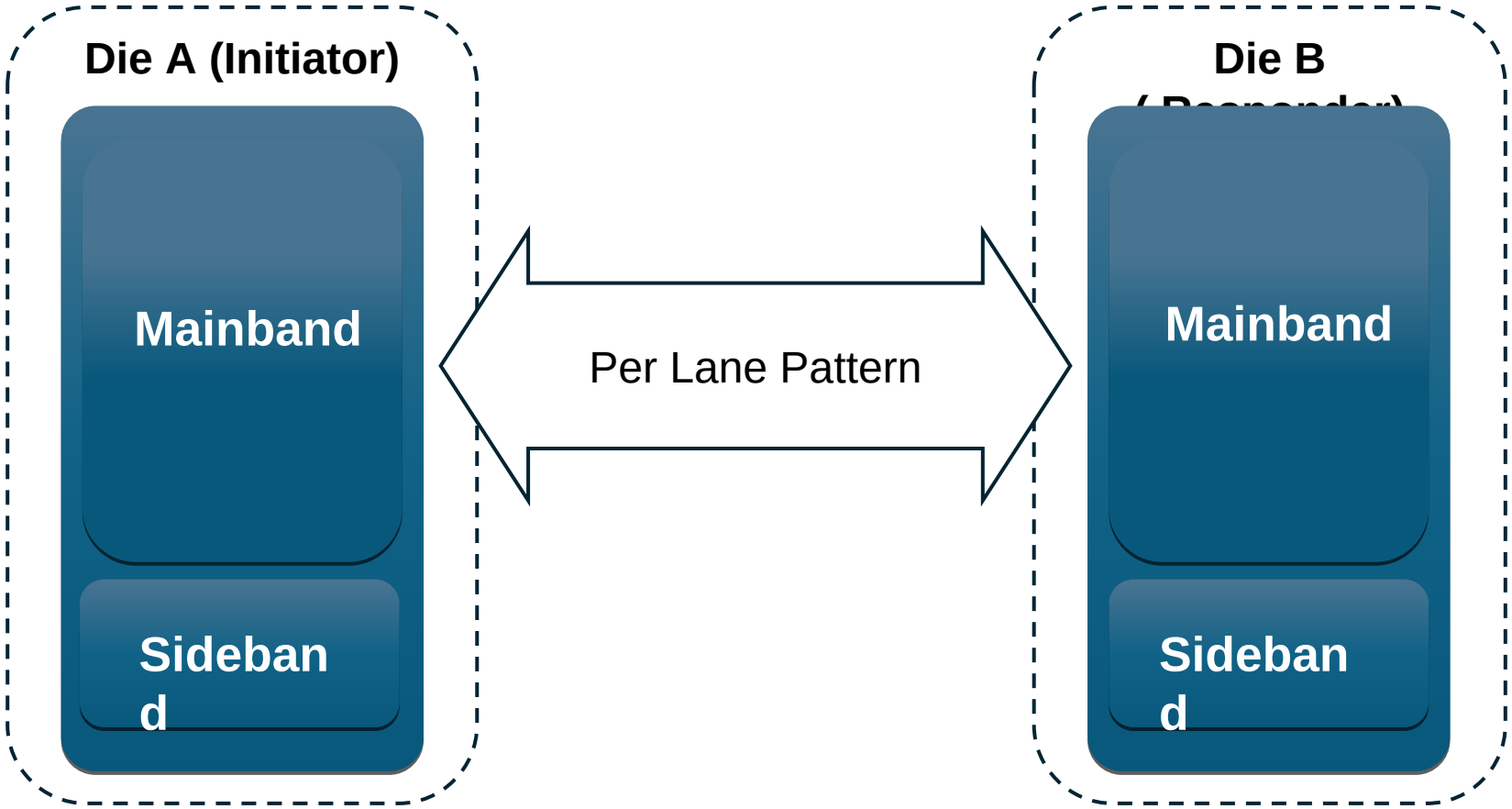

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

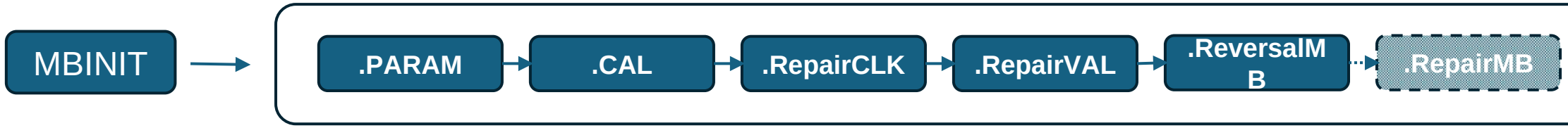

Die A (Initiator)
Mainband
Sideband
{clear_error_req}
msg
Die B
Mainband
Sideband

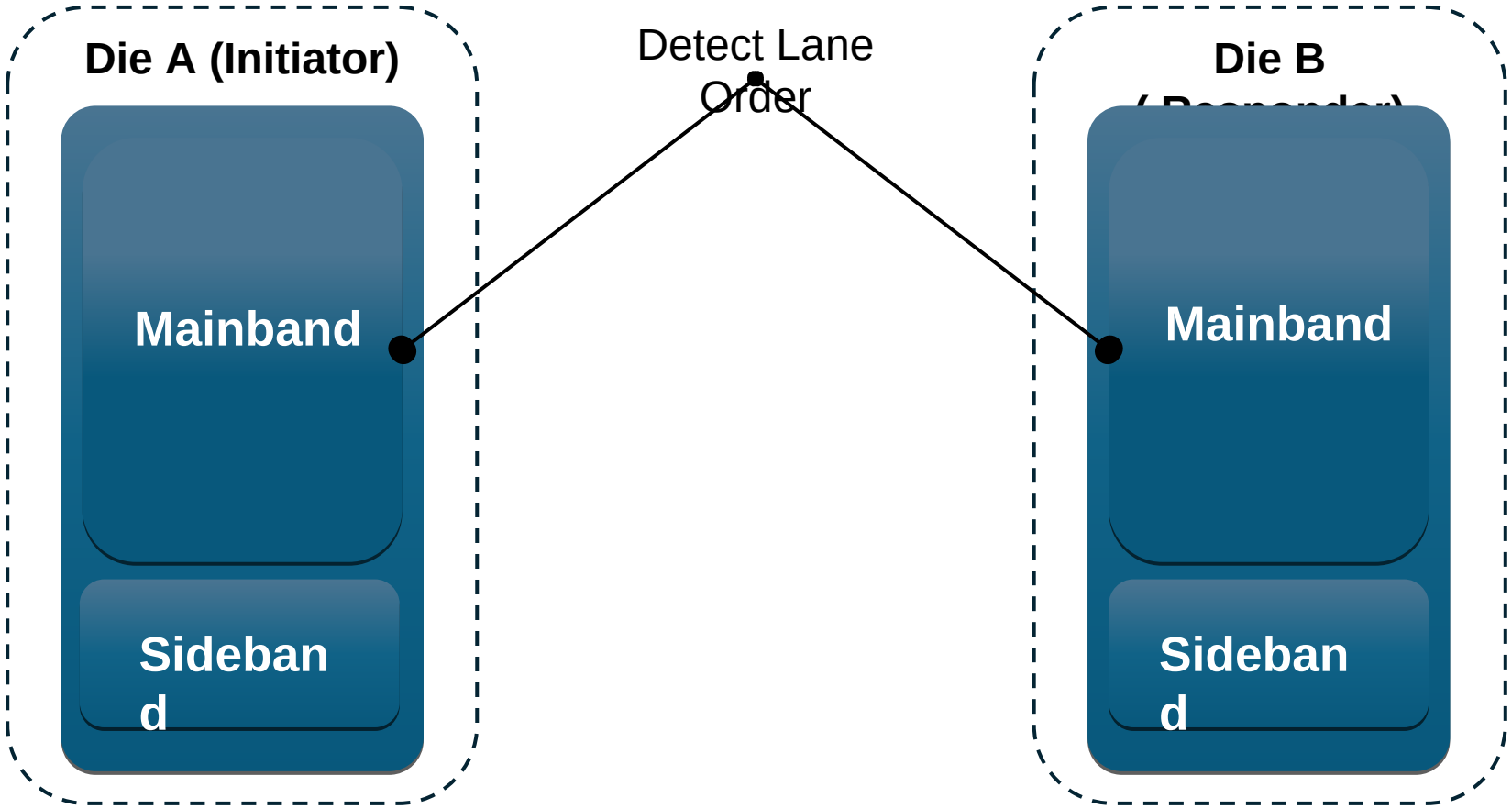

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

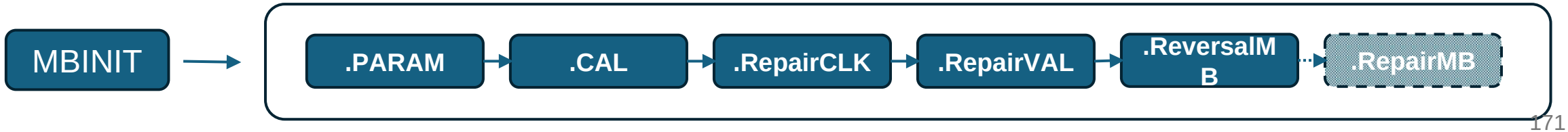

Die A (Initiator)
Clear Error
Die B
Mainband
Sideband
{clear_error_resp} msg
Mainband
Sideband

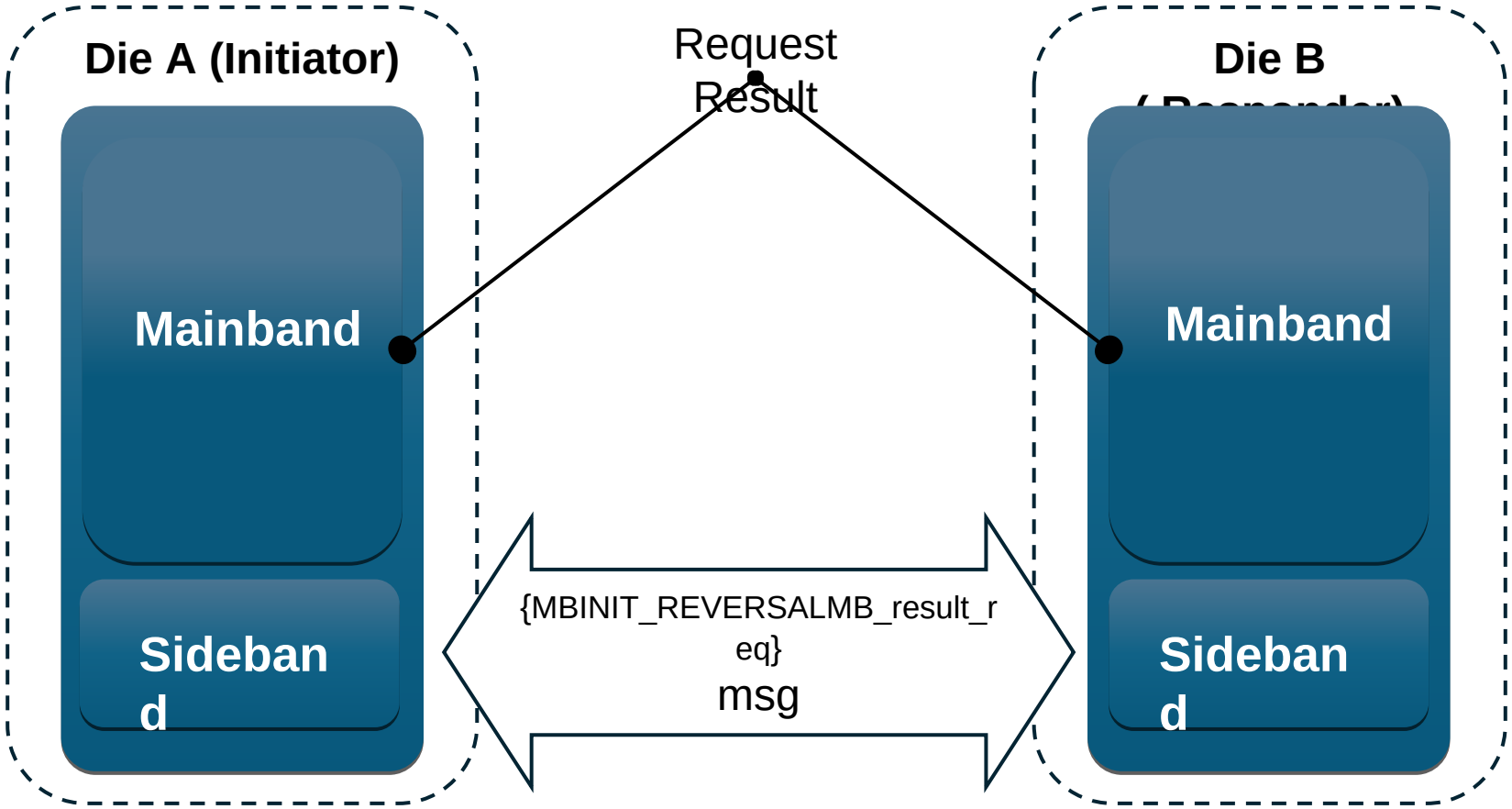

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

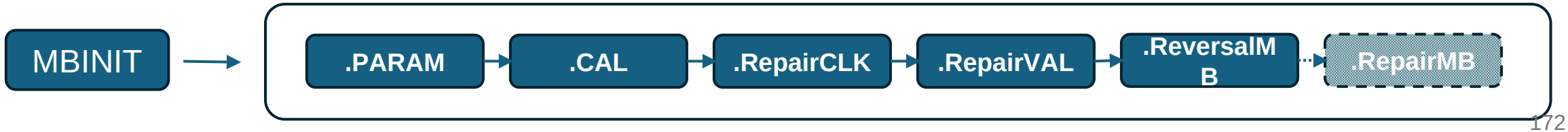

Die A (Initiator)
Mainband
Sideband
Per Lane Pattern
Die B
Mainband
Sideband

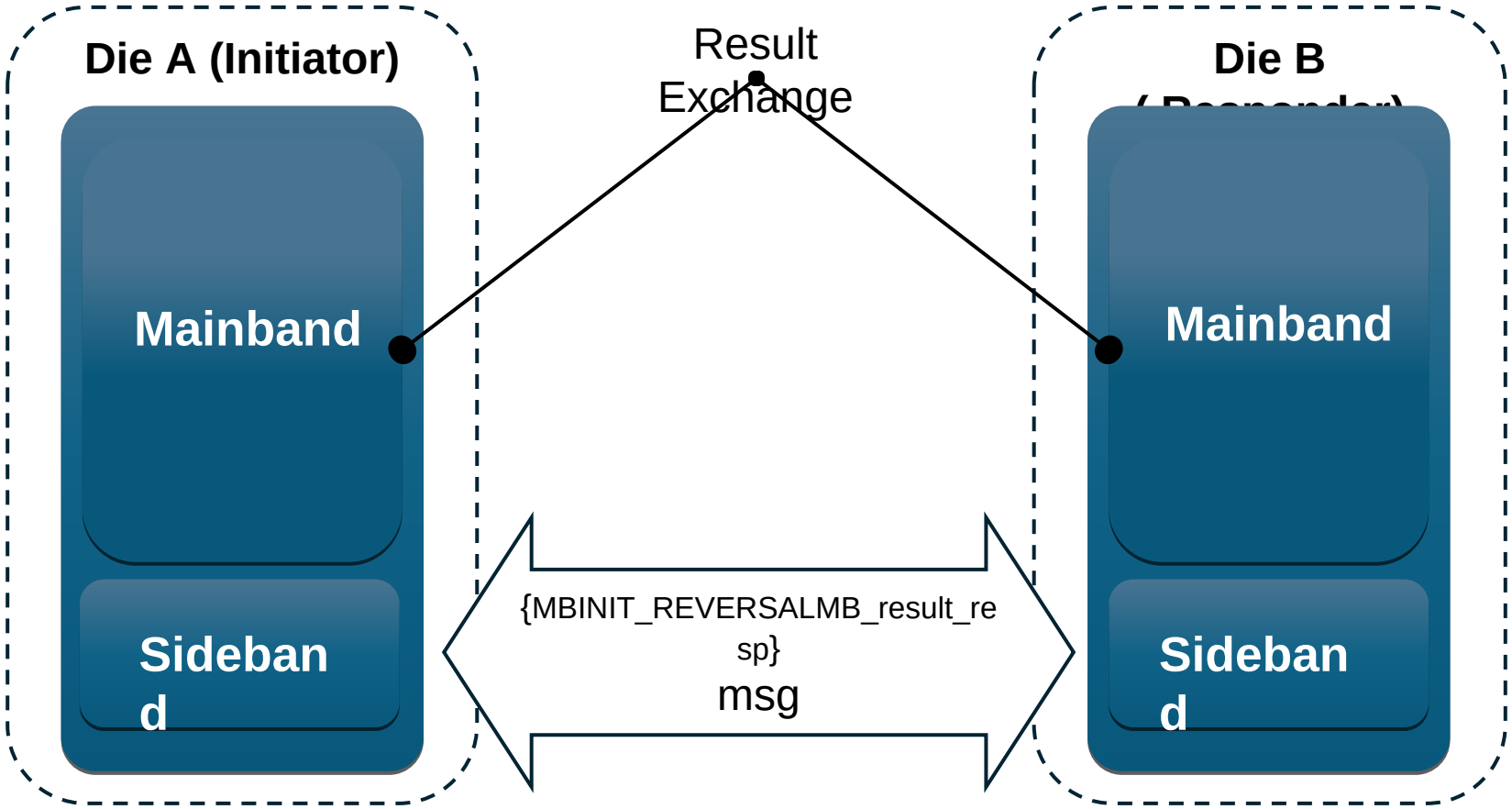

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

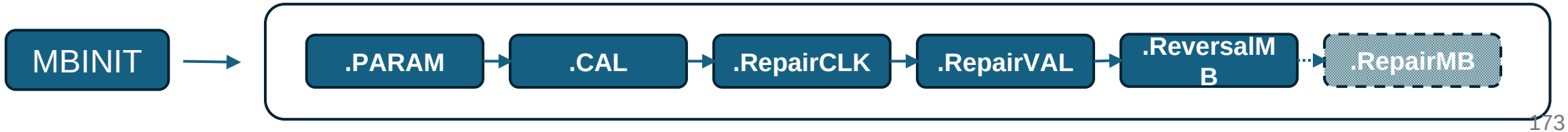

Die A (Initiator)
Detect Lane Order
Die B
Mainband
Sideband
Mainband
Sideband

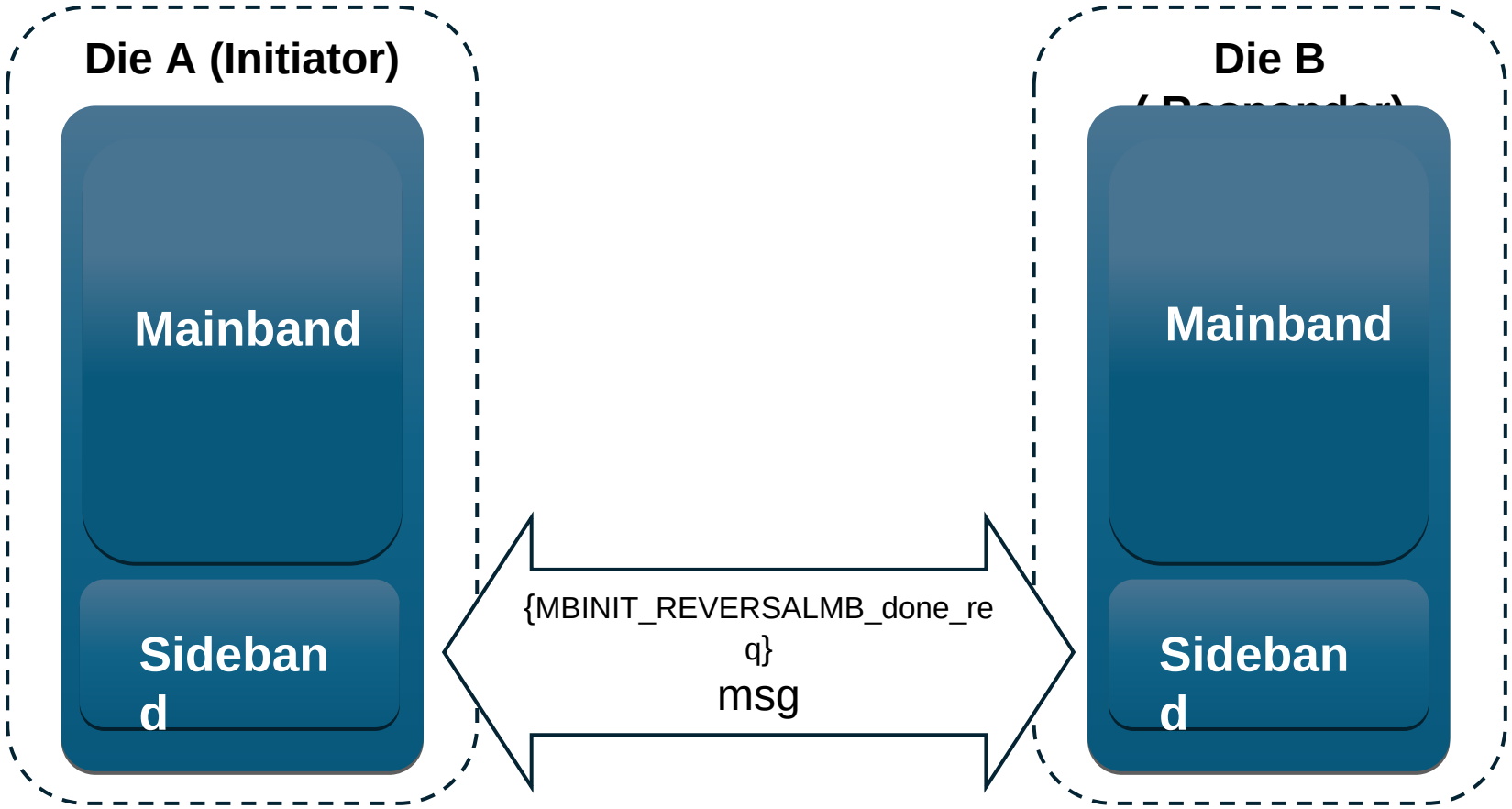

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

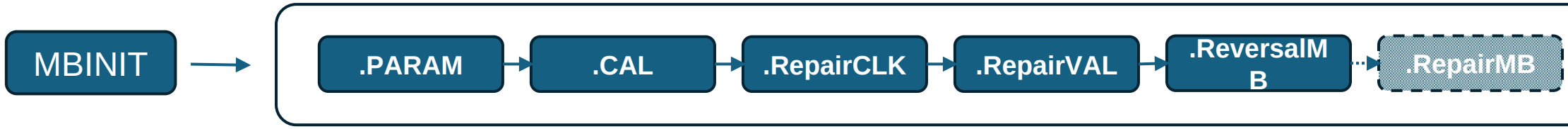

Die A (Initiator)
Request
Result
Die B
Mainband
Sideband
{MBINIT_REVERSALMB_result_req}
msg
Mainband
Sideband

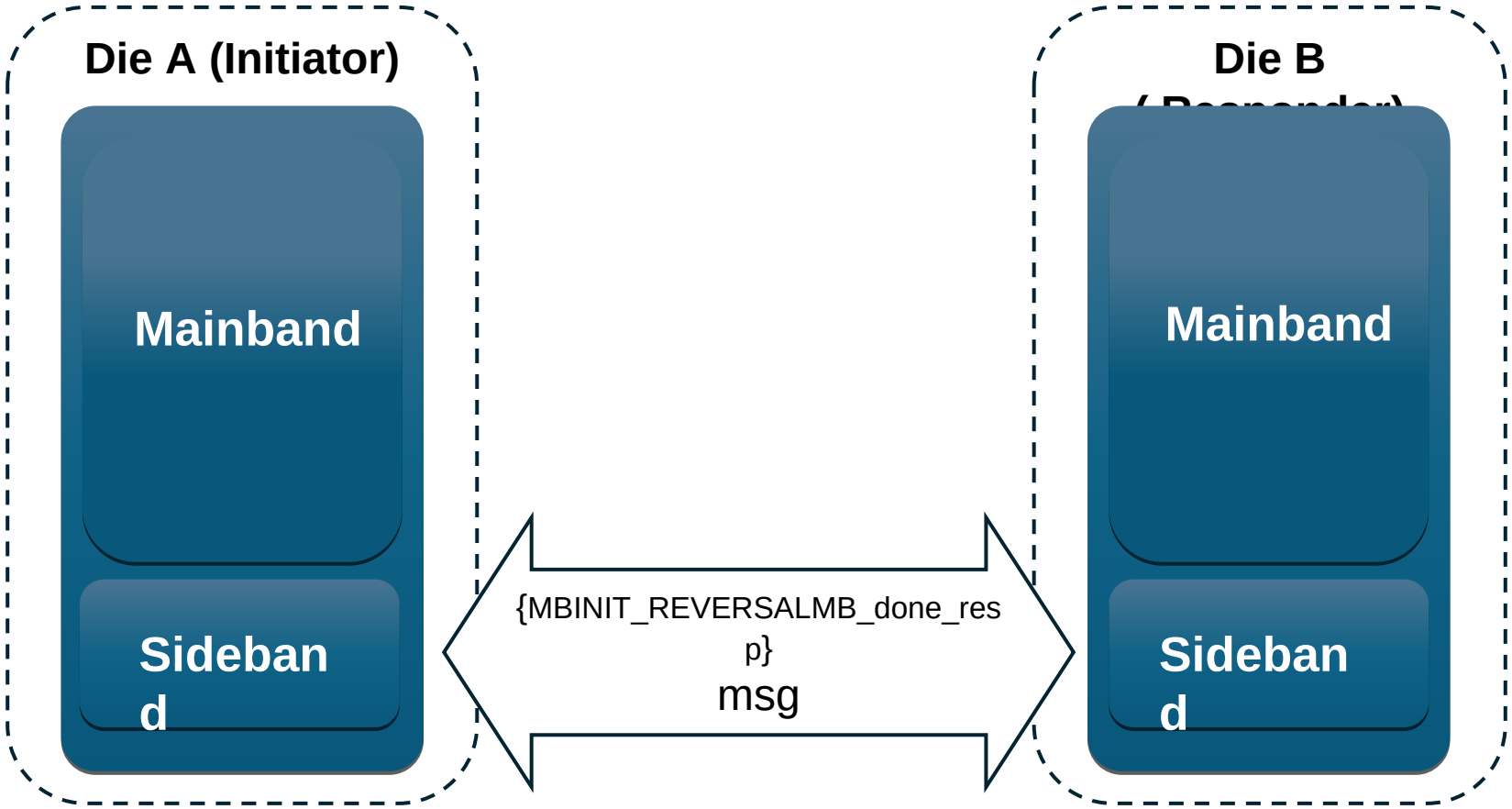

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

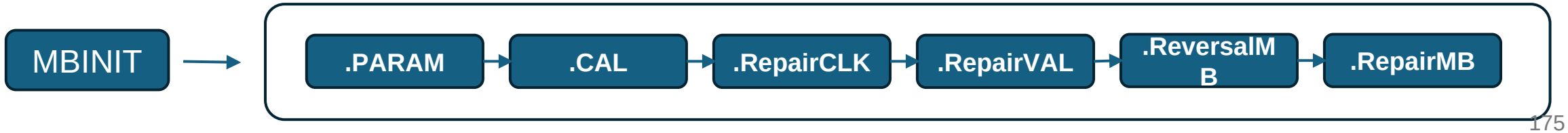

Die A (Initiator)
Result Exchange
Die B
Mainband
Sideband
{MBINIT_REVERSALMB_result_resp}
msg
Mainband
Sideband

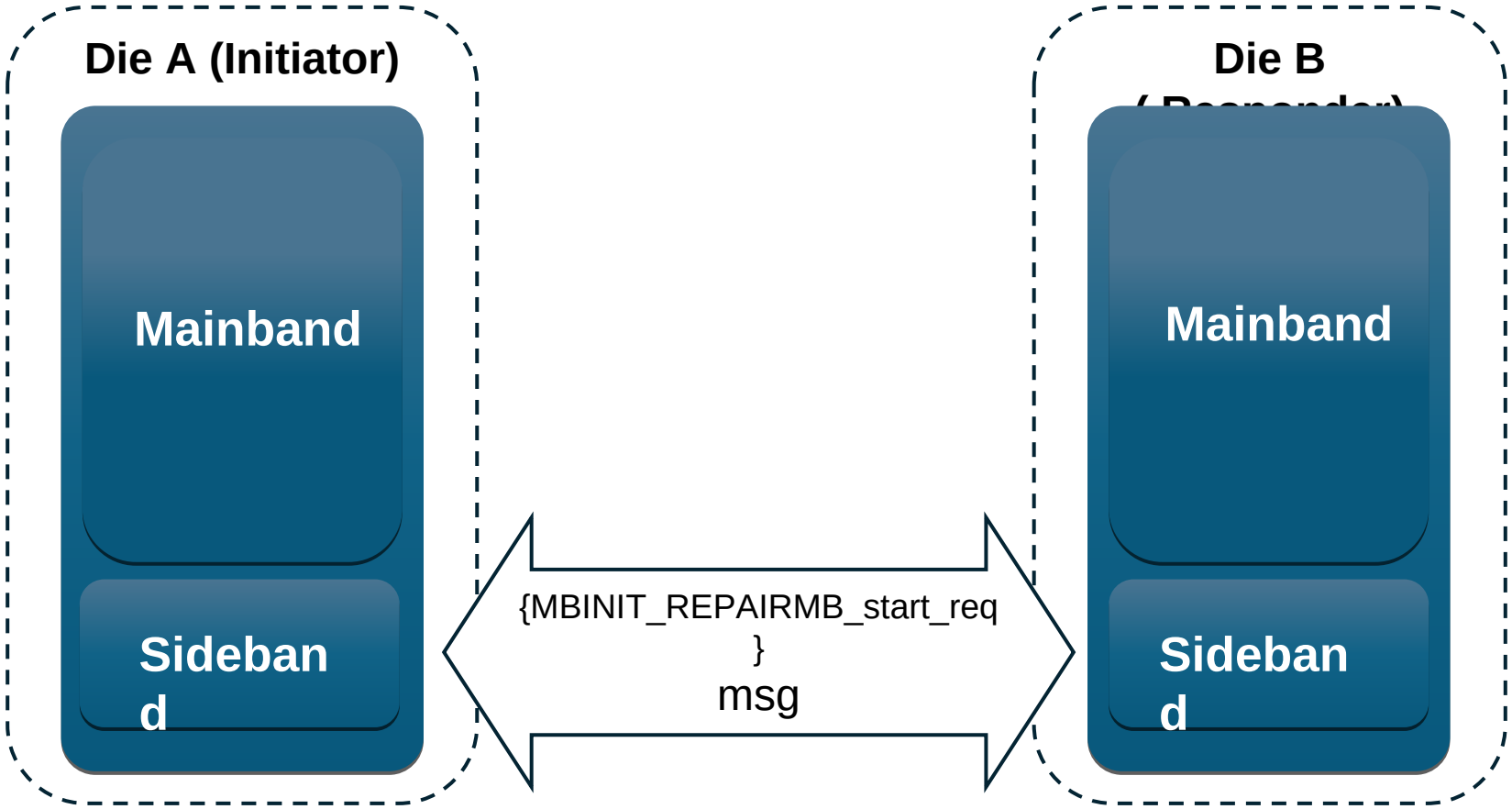

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

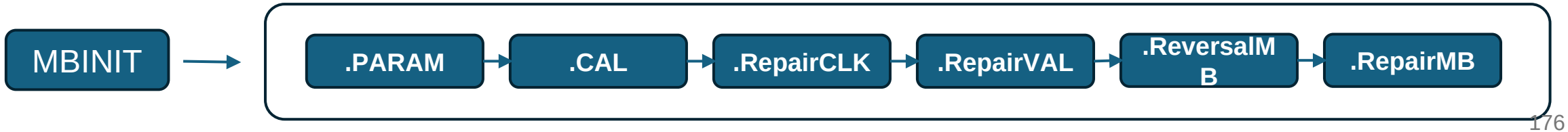

Die A (Initiator)
Mainband
Sideband
{MBINIT_REVERSALMB_done_req}
msg
Die B
Mainband
Sideband

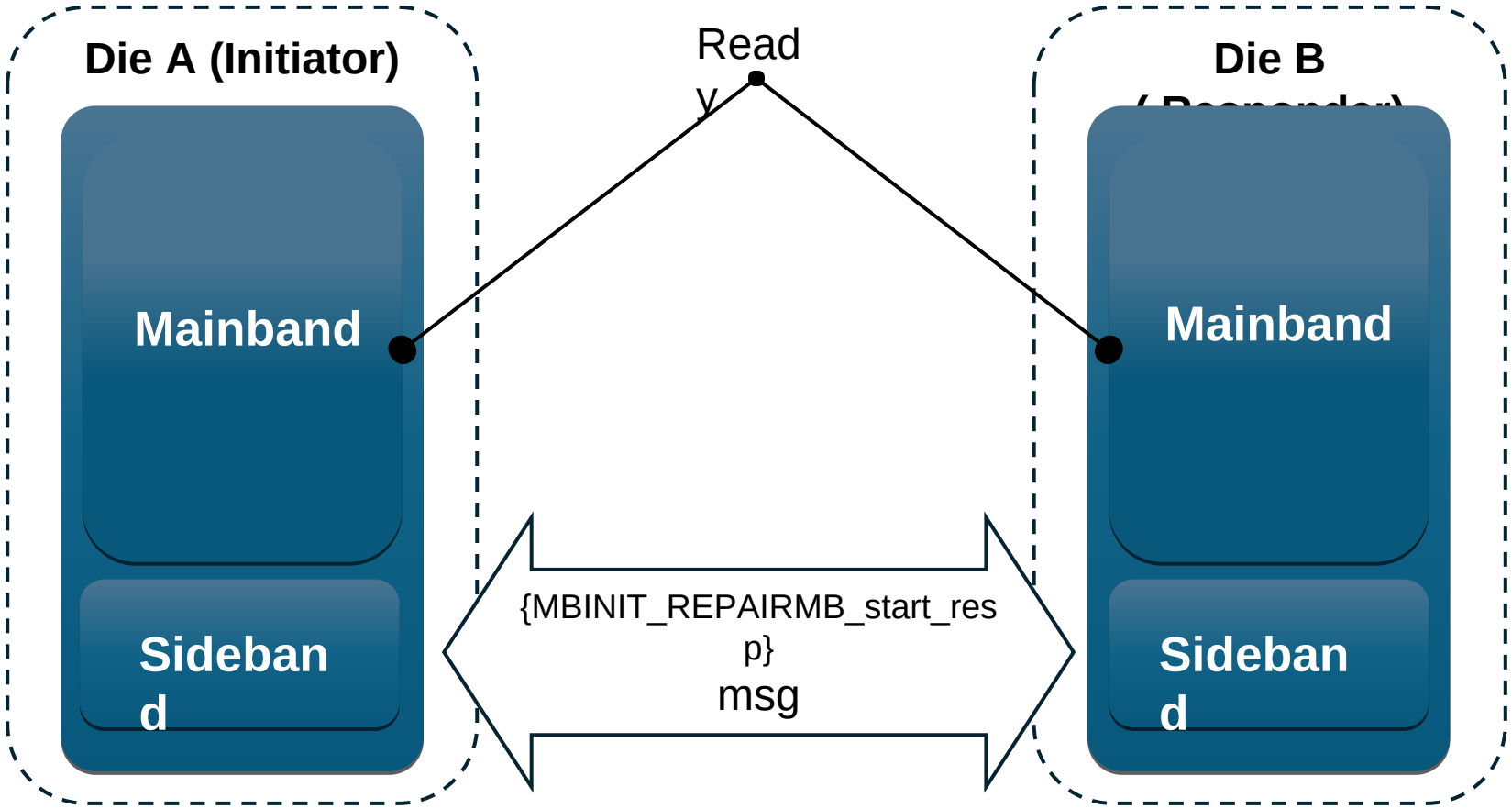

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

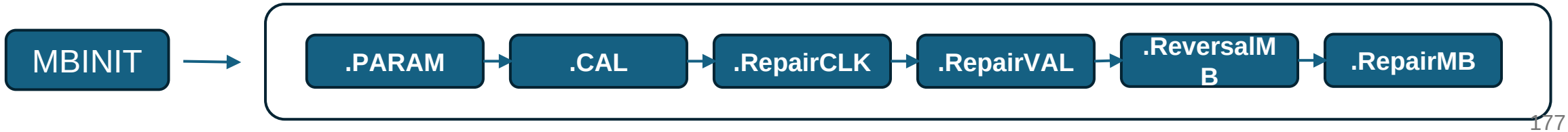

Die A (Initiator)
Mainband
Sideband
{MBINIT_REVERSALMB_done_resp}
msg
Die B
Mainband
Sideband

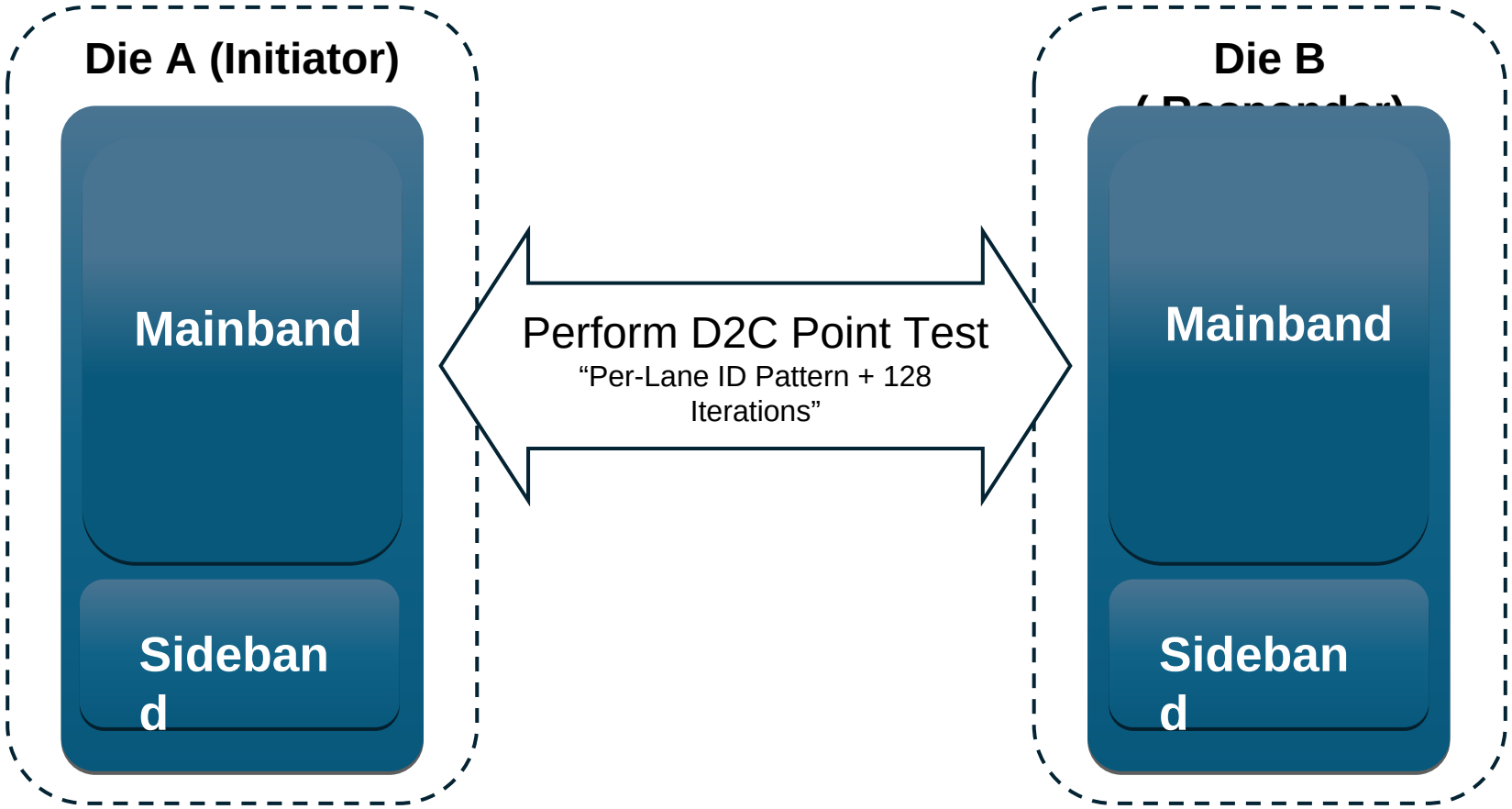

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

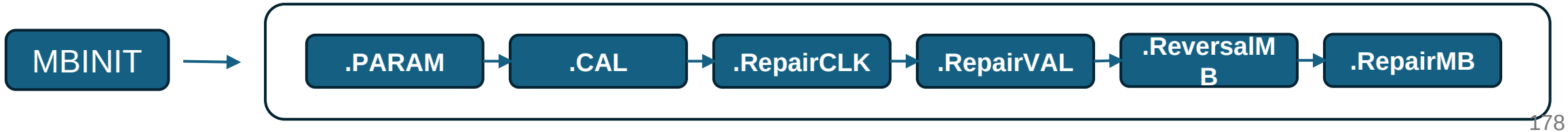

Die A (Initiator)
Mainband
Sideband
{MBINIT_REPAIRMB_start_req
}
msg
Die B
Mainband
Sideband

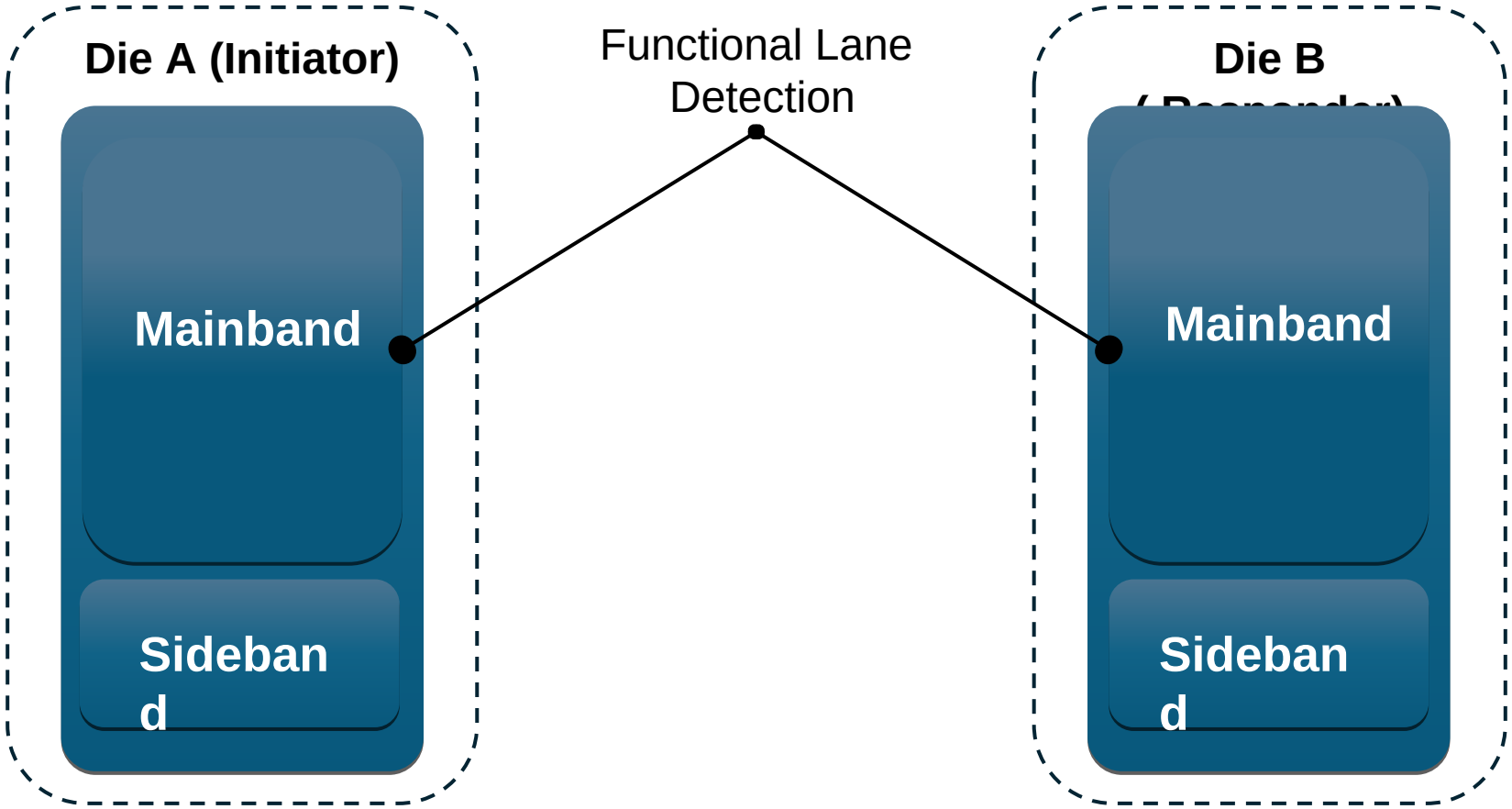

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

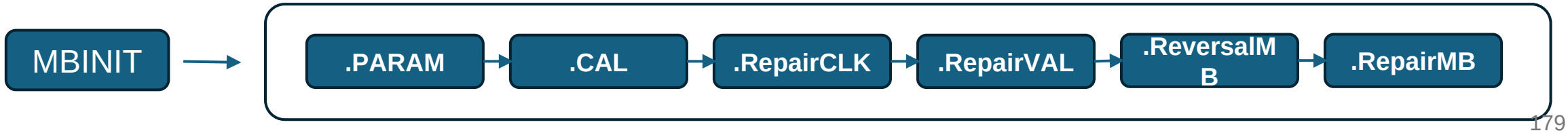

Die A (Initiator)
Ready
Die B
Mainband
Sideband
{MBINIT_REPAIRMB_start_resp}
msg
Mainband
Sideband

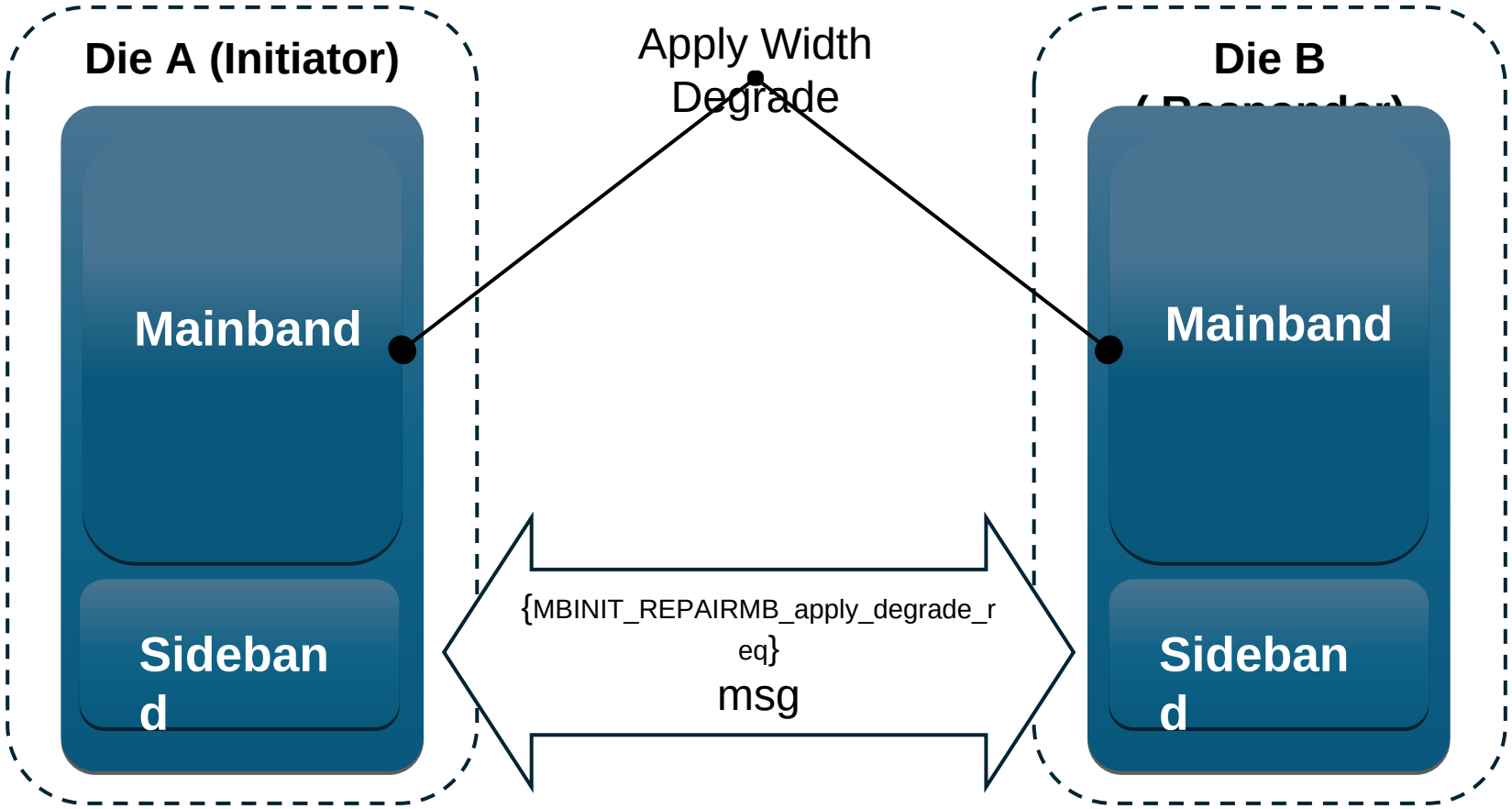

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

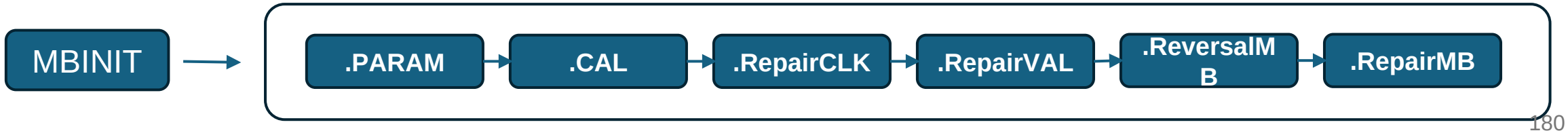

Die A (Initiator)
Mainband
Sideband
Perform D2C Point Test
“Per-Lane ID Pattern + 128 Iterations”
Die B
Mainband
Sideband

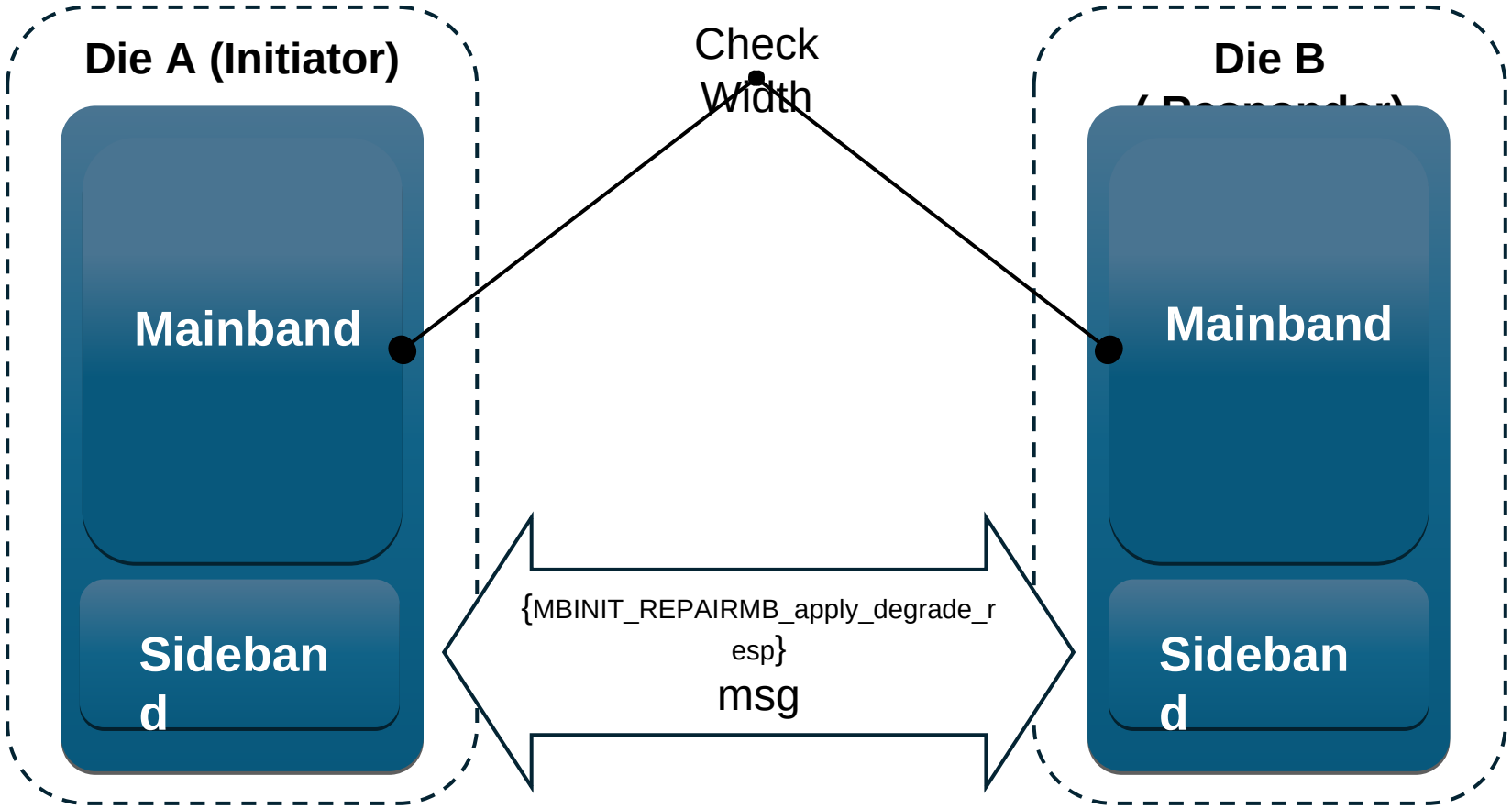

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

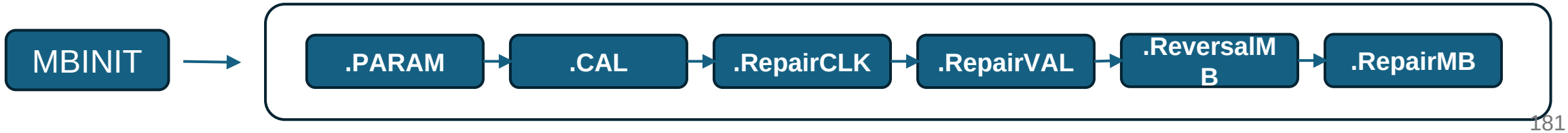

Die A (Initiator)
Functional Lane Detection
Die B
Mainband
Sideband
Mainband
Sideband

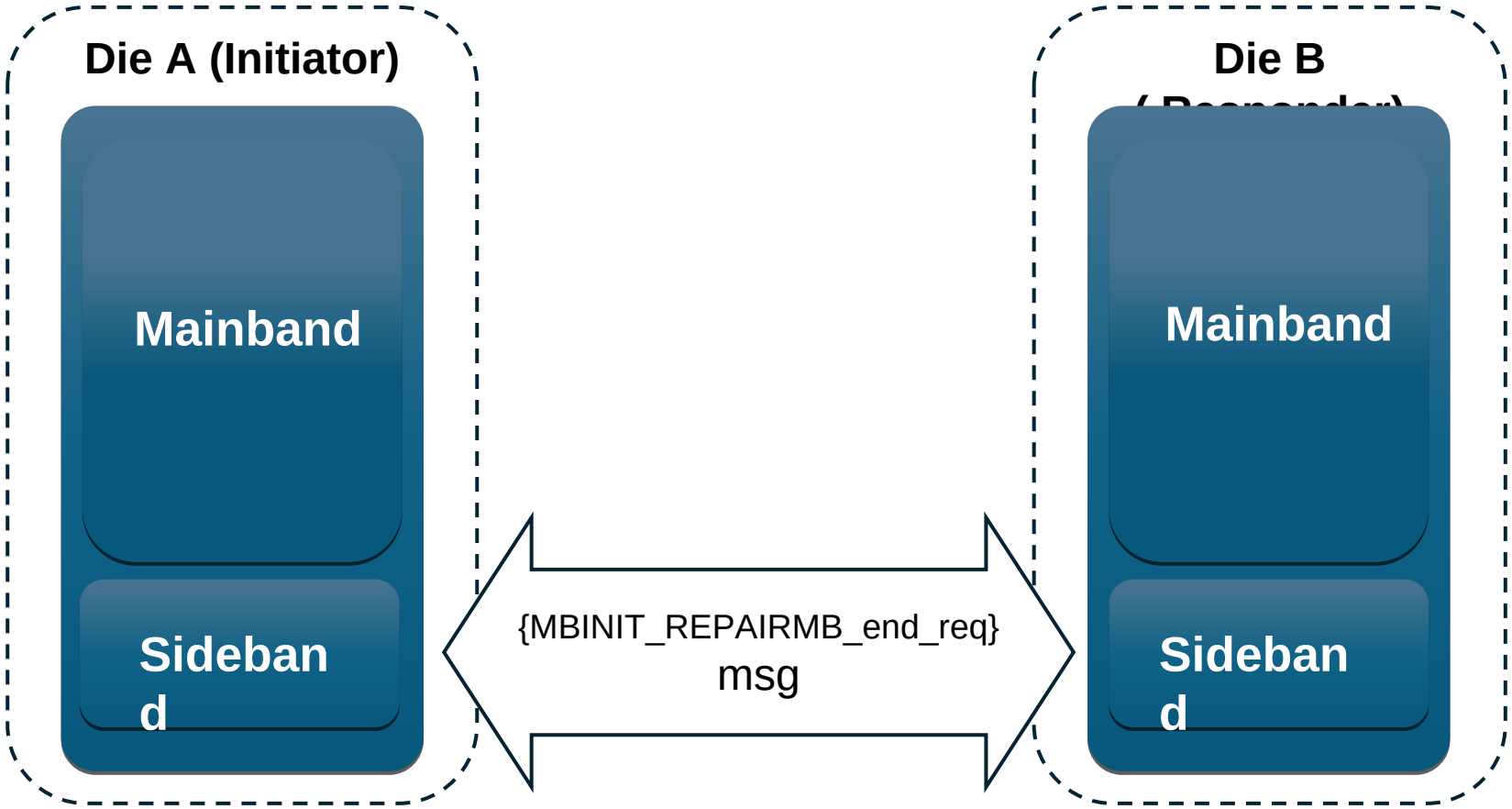

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

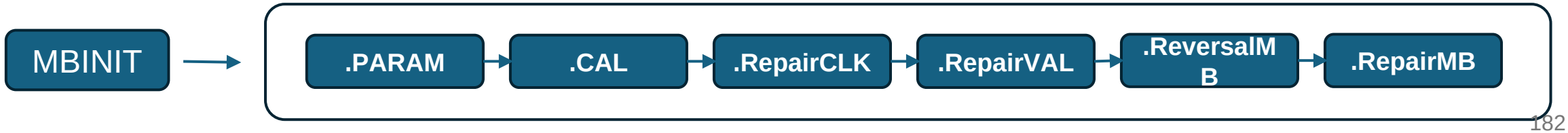

Die A (Initiator)
Apply Width Degrade
Die B
Mainband
Sideband
{MBINIT_REPAIRMB_apply_degrade_req}
msg
Mainband
Sideband

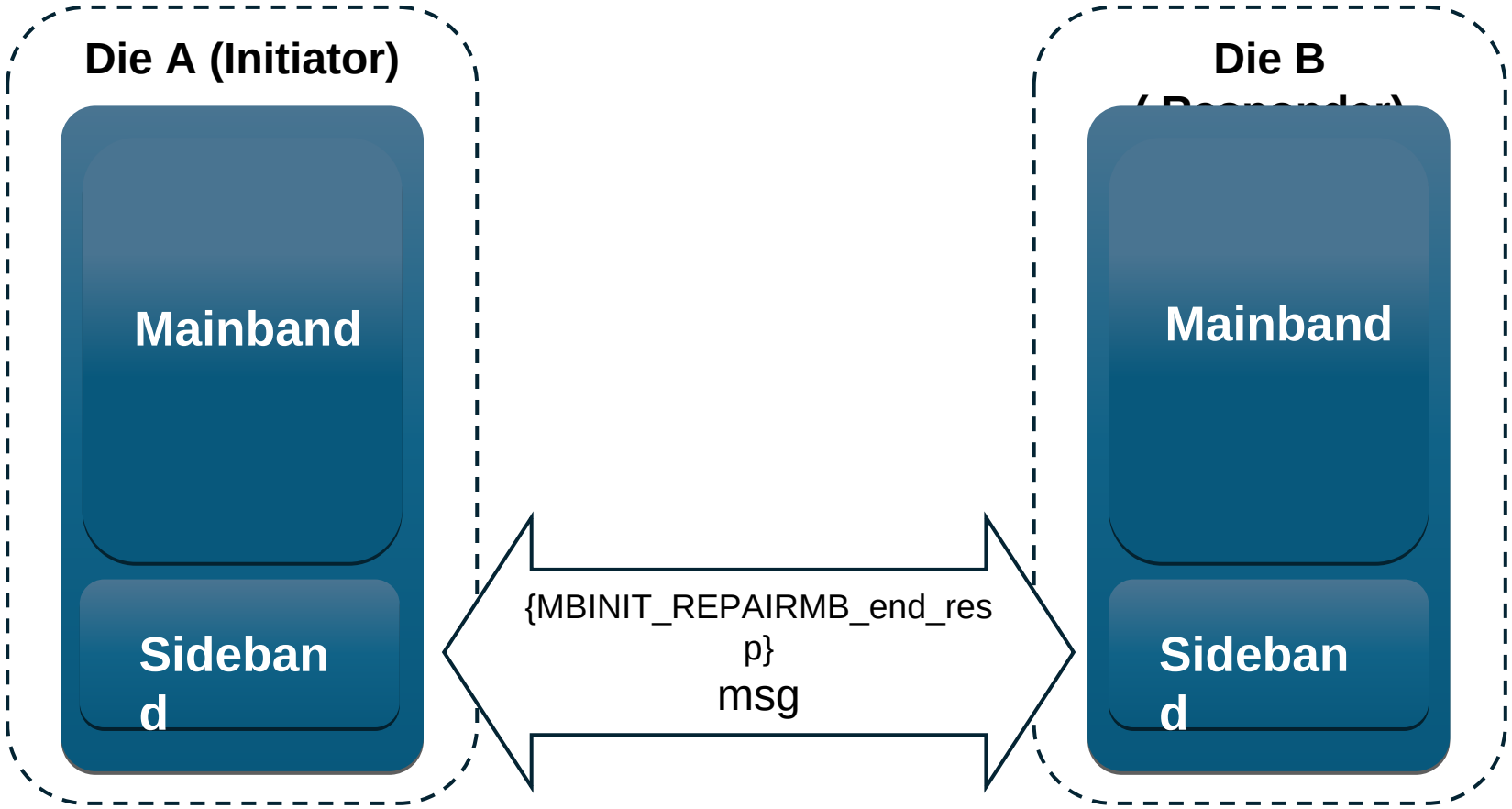

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

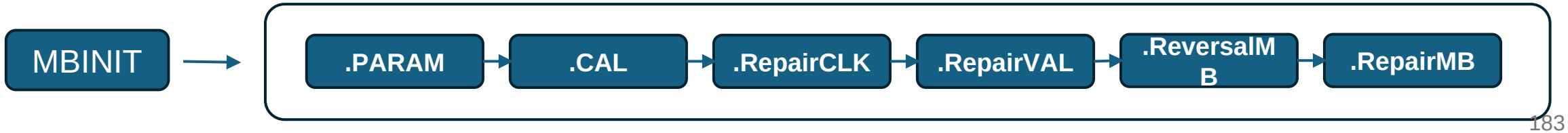

Die A (Initiator)
Check Width
Die B
Mainband
Sideband
{MBINIT_REPAIRMB_apply_degrade_resp}
msg
Mainband
Sideband

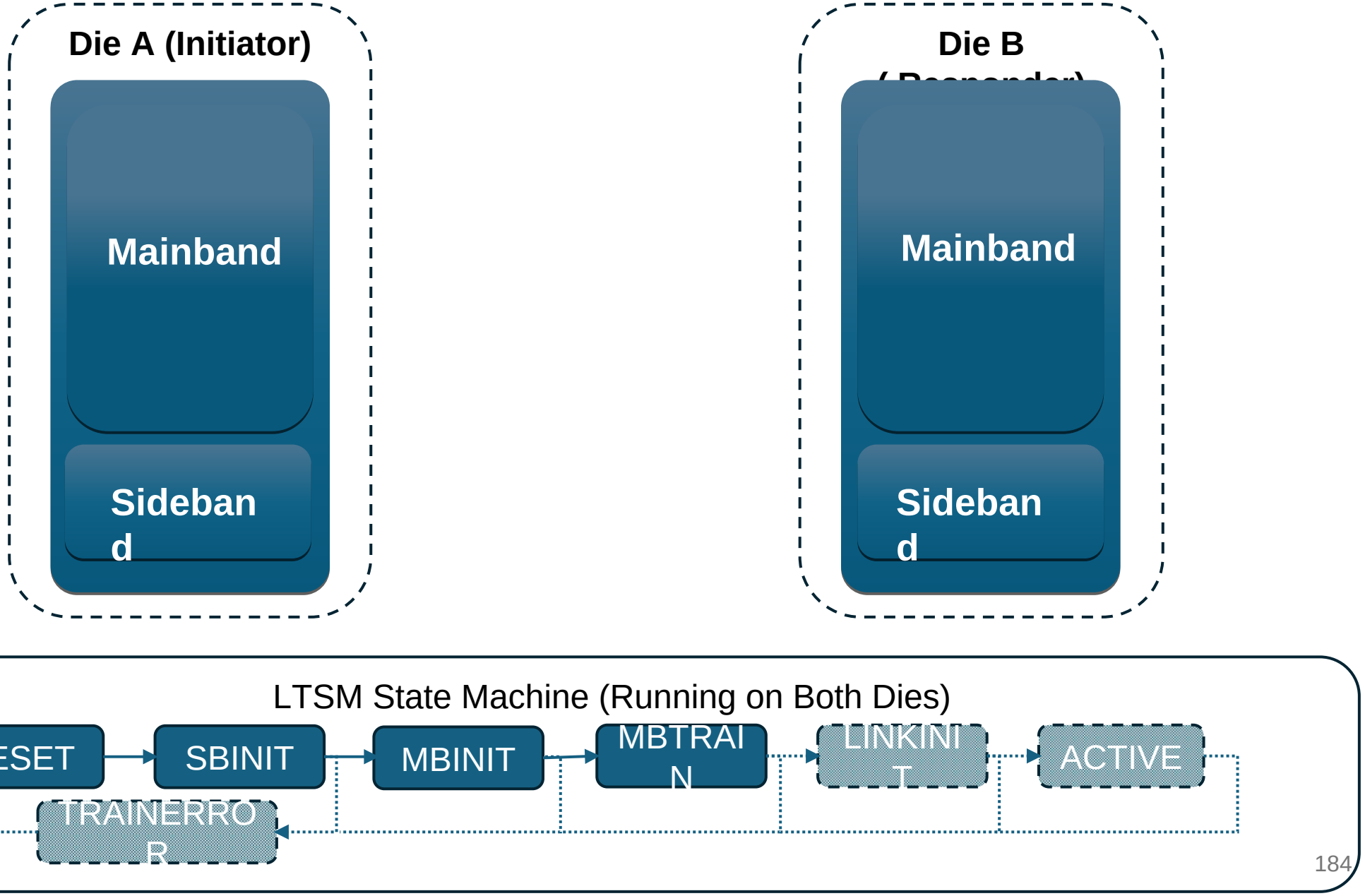

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

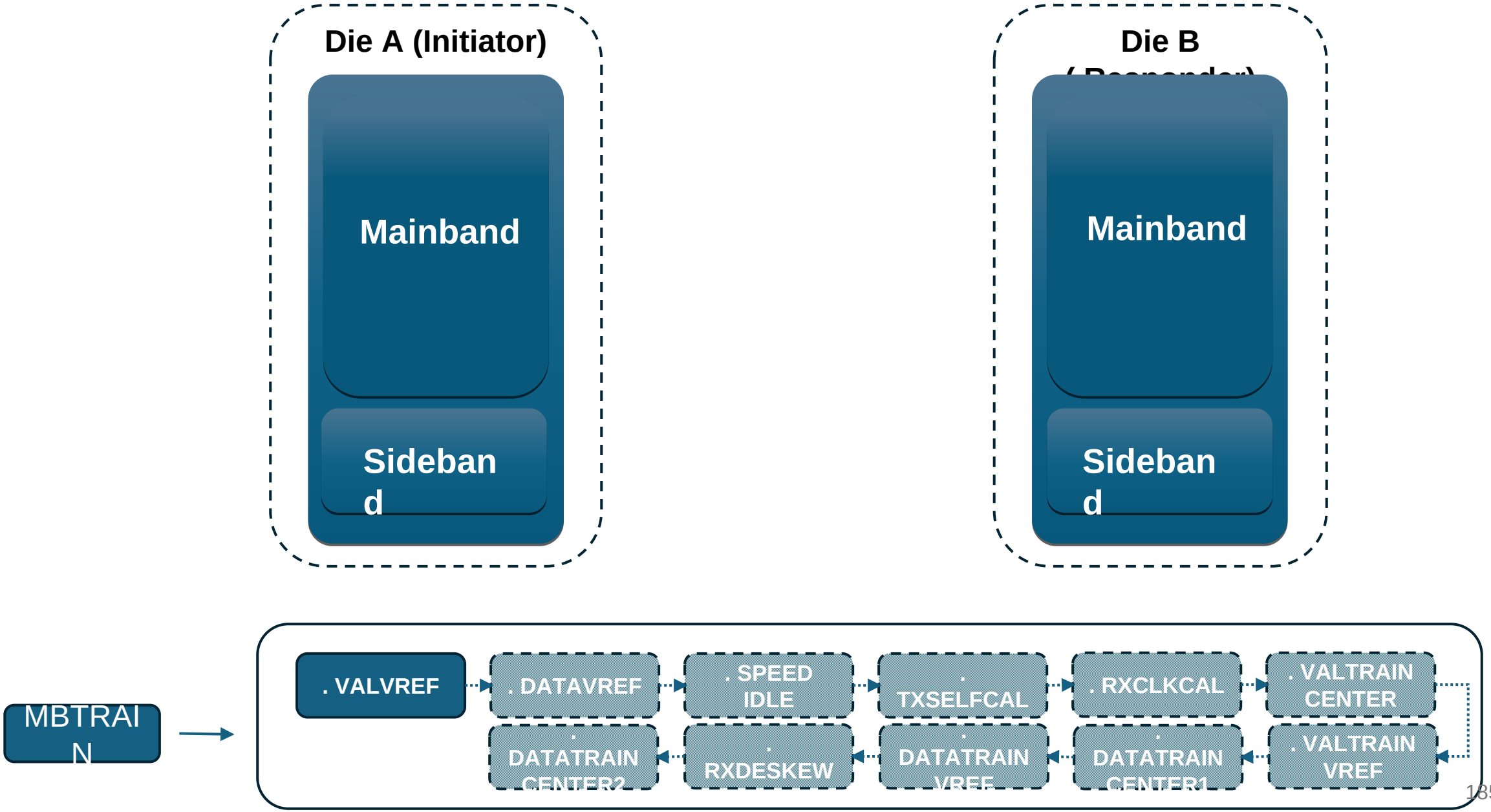

Die A (Initiator)
Mainband
Sideband
{MBINIT_REPAIRMB_end_req}
msg
Die B
Mainband
Sideband

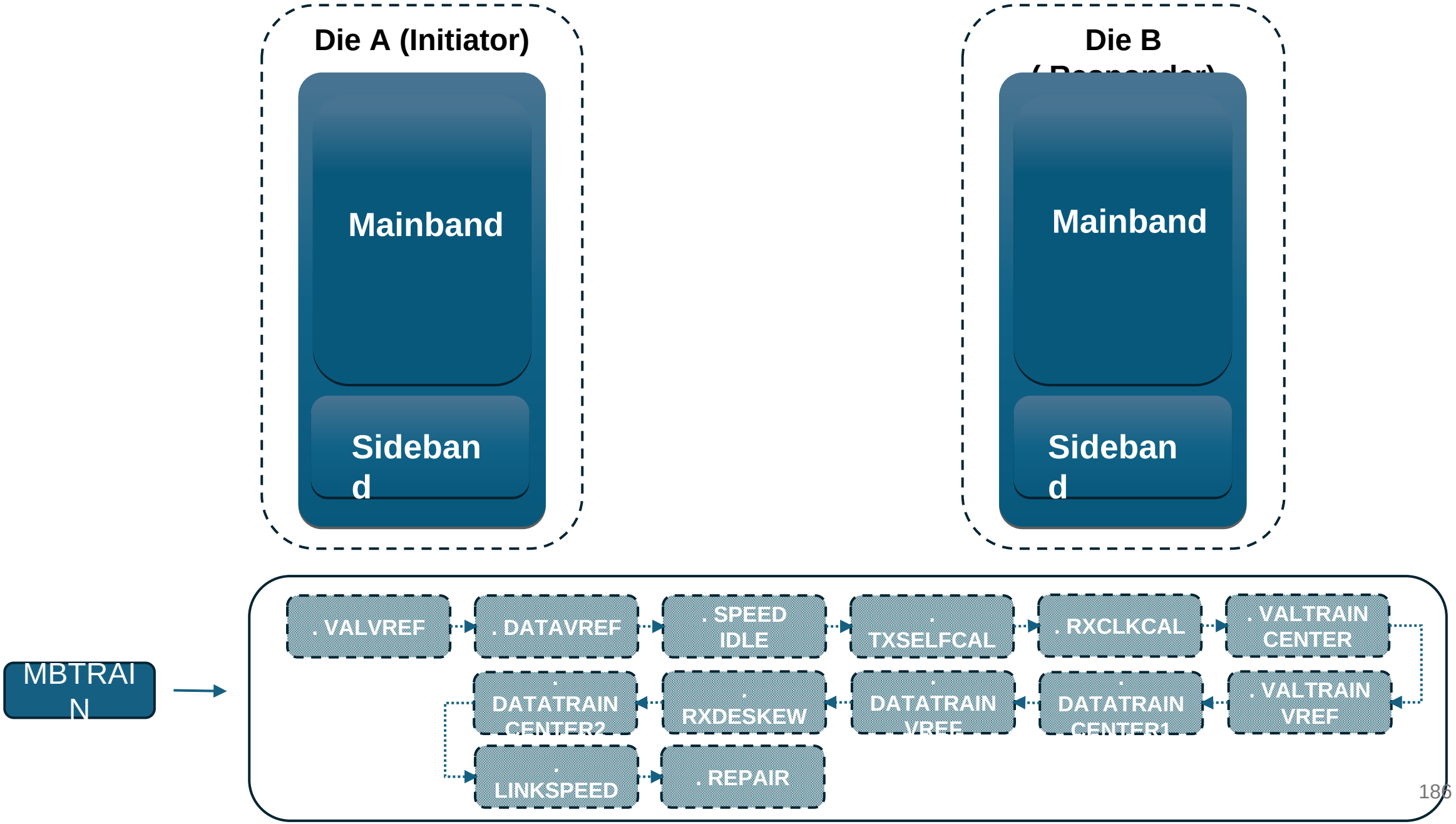

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

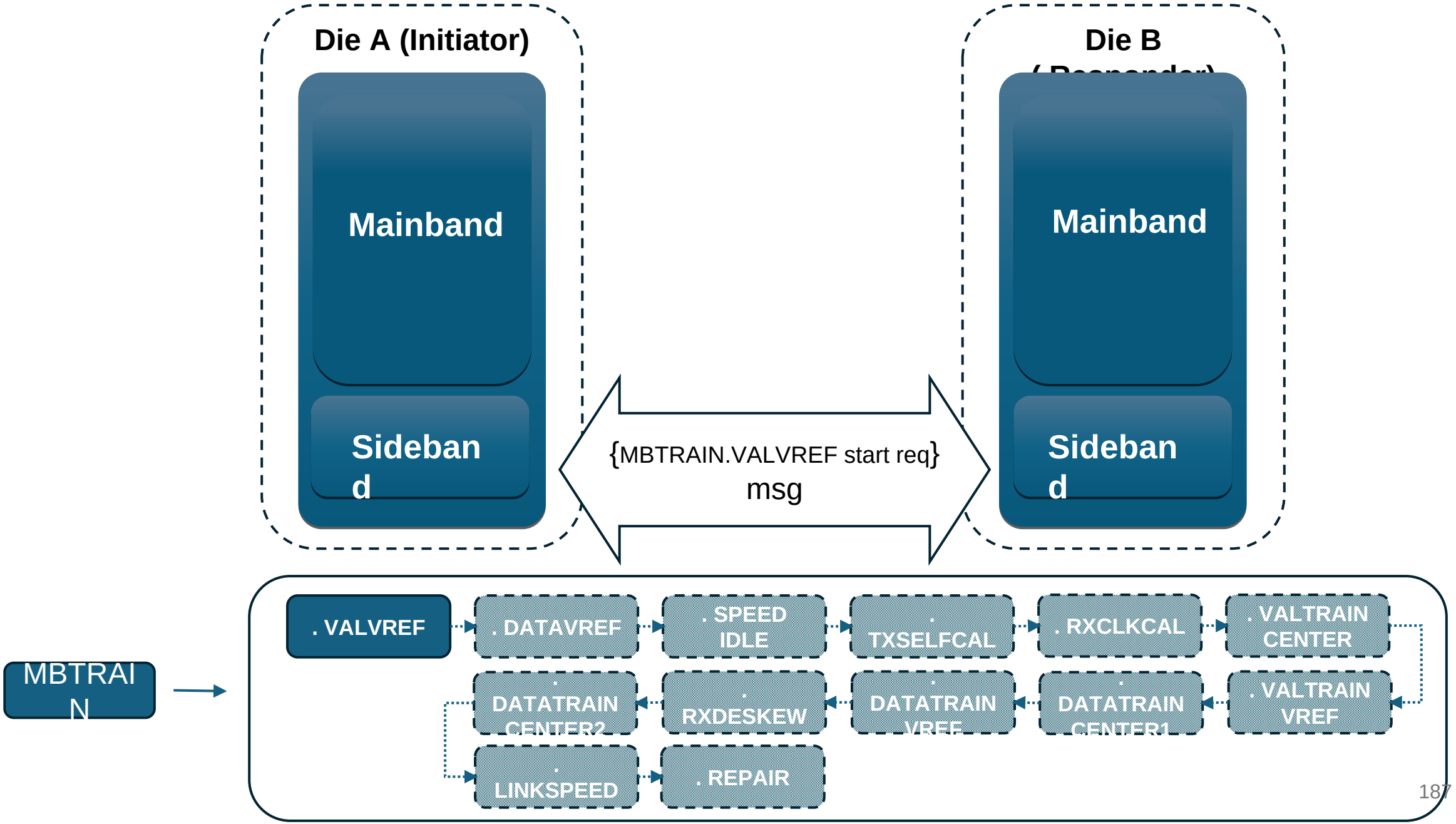

Die A (Initiator)
Mainband
Sideband
{MBINIT_REPAIRMB_end_resp}
msg
Die B
Mainband
Sideband

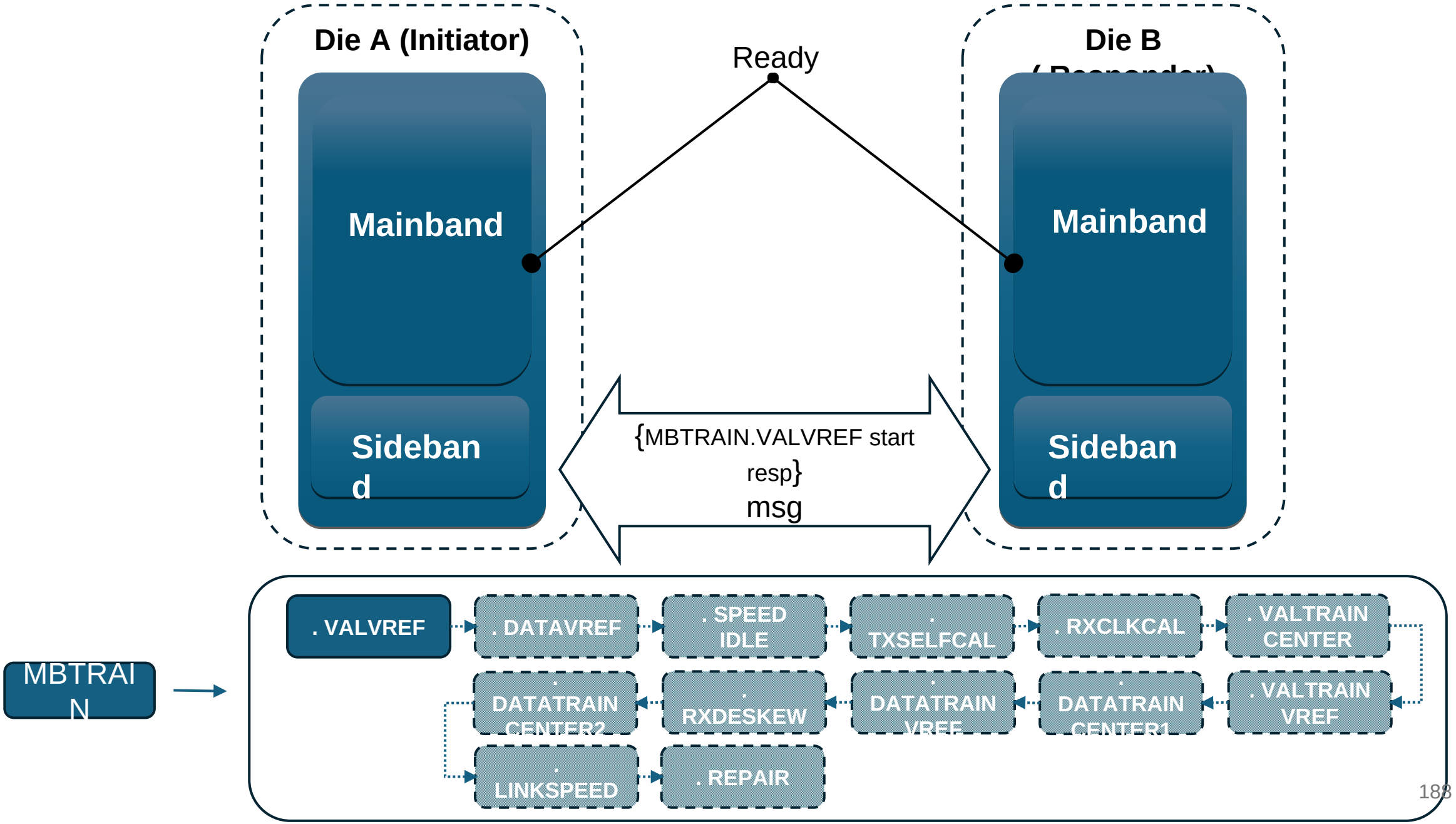

MBINIT
.PARAM
.CAL
.RepairCLK
.RepairVAL
.ReversalMB
.RepairMB

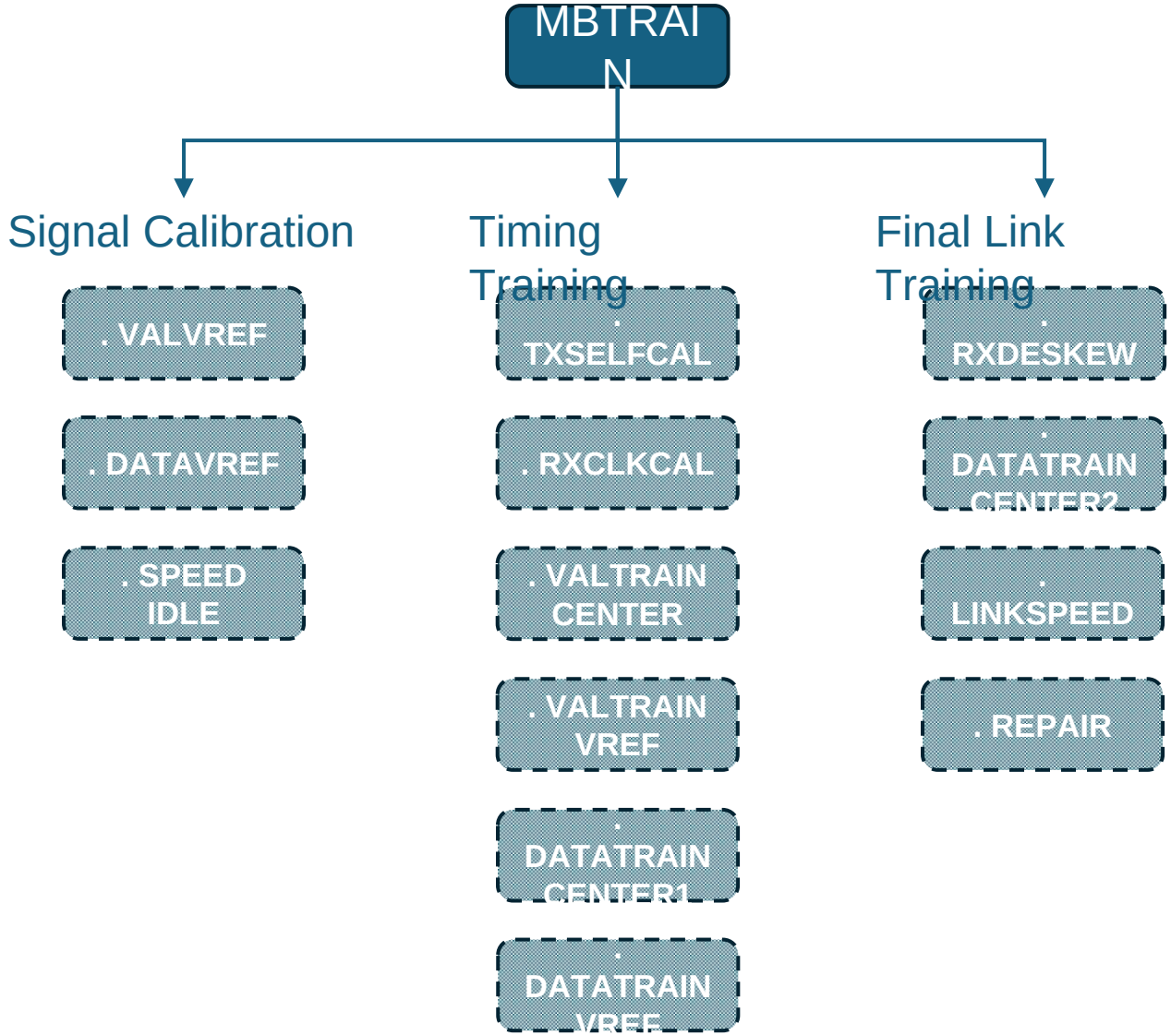

Die A (Initiator)
Mainband
Sideband
Die B
Mainband
Sideband
LTSM State Machine (Running on Both Dies)
Training Trigger
RESET
SBINIT
MBINIT
MBTRAIN
LINKINIT
ACTIVE
TRAINERROR


Die A (Initiator)
Mainband
Sideband
Die B
Mainband
Sideband
MBTRAIN
. VALVREF
. DATAVREF
. SPEED IDLE
. TXSELFCAL
. RXCLKCAL
. VALTRAIN CENTER
. DATATRAIN CENTER2
. RXDESKEW
. DATATRAIN VREF
. DATATRAIN CENTER1
. VALTRAIN VREF


Die A (Initiator)
Mainband
Sideband
Die B
Mainband
Sideband
MBTRAIN
. VALVREF
. DATAVREF
. SPEED IDLE
. TXSELFCAL
. RXCLKCAL
. VALTRAIN CENTER
. VALTRAIN VREF
. DATATRAIN CENTER1
. DATATRAIN VREF
. RXDESKEW
. DATATRAIN CENTER2
. LINKSPEED
. REPAIR


Die A (Initiator)
Mainband
Sideband
{MBTRAIN.VALVREF start req}
msg
Die B
Mainband
Sideband
MBTRAIN
. VALVREF
. DATAVREF
. SPEED IDLE
. TXSELFCAL
. RXCLKCAL
. VALTRAIN CENTER
. VALTRAIN VREF
. DATATRAIN CENTER1
. DATATRAIN VREF
. RXDESKEW
. DATATRAIN CENTER2
. LINKSPEED
. REPAIR

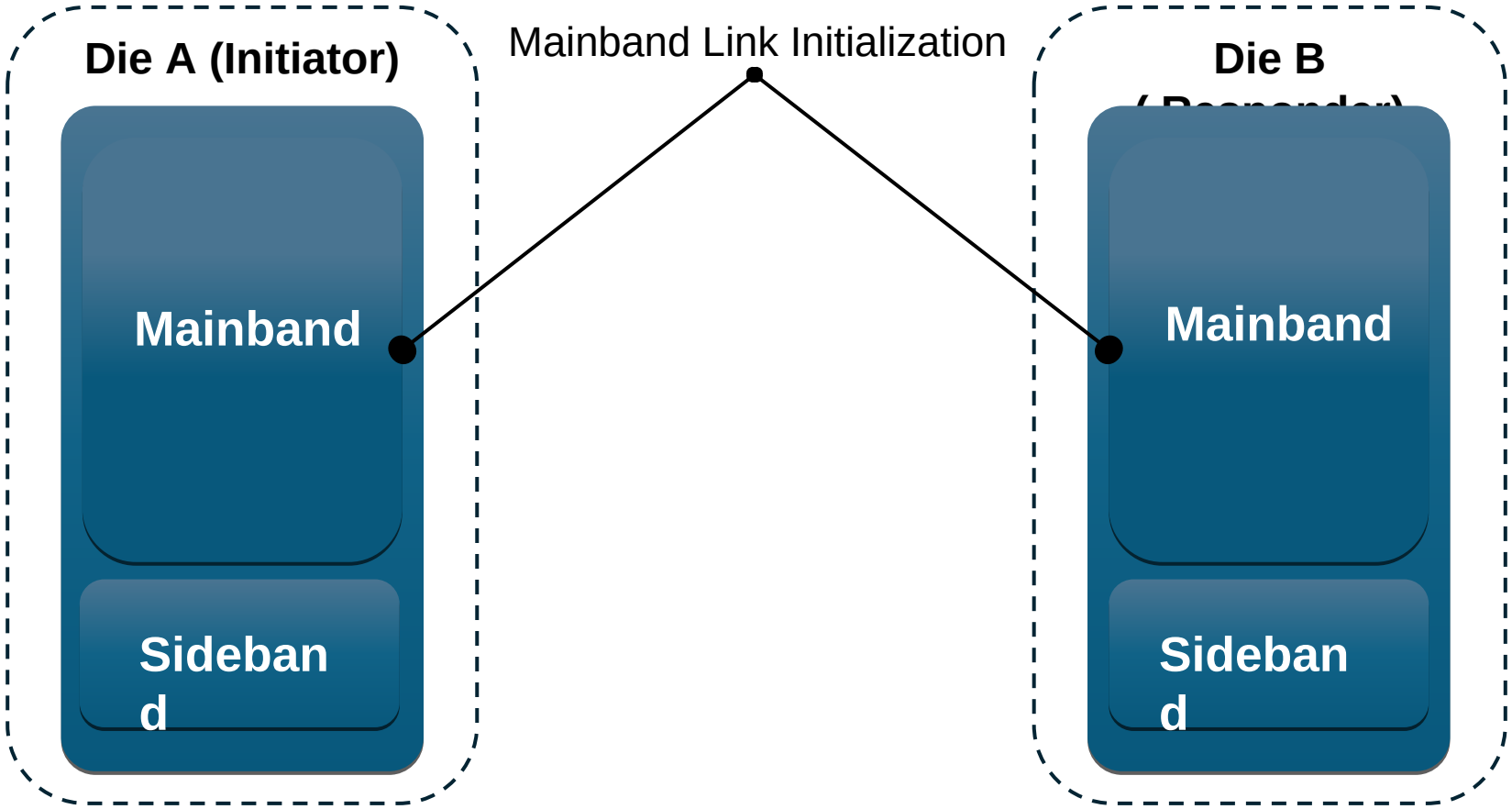

Die A (Initiator)
Mainband
Sideband
Ready
{MBTRAIN.VALVREF start resp}
msg
Die B
Mainband
Sideband
MBTRAIN
. VALVREF
. DATAVREF
. SPEED IDLE
. TXSELFCAL
. RXCLKCAL
. VALTRAIN CENTER
. VALTRAIN VREF
. DATATRAIN CENTER1
. DATATRAIN VREF
. RXDESKEW
. DATATRAIN CENTER2
. LINKSPEED
. REPAIR

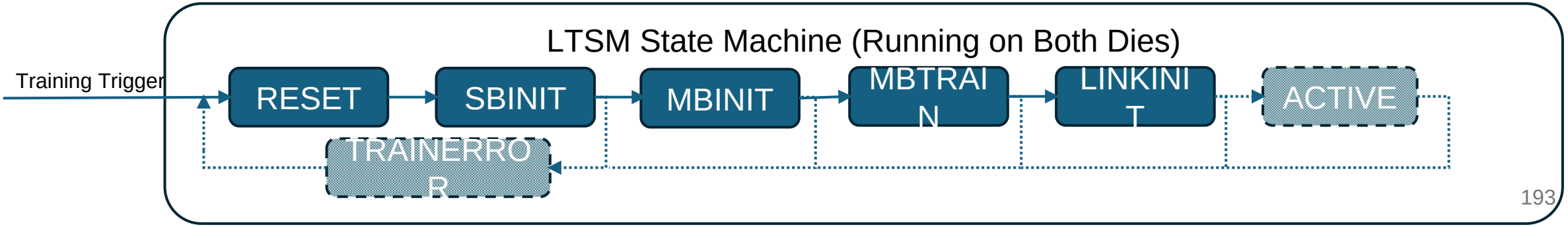

MBTRAIN
Signal Calibration
. VALVREF
. DATAVREF
. SPEED IDLE
Timing Training
. TXSELFCAL
. RXCLKCAL
. VALTRAIN CENTER
. VALTRAIN VREF
. DATATRAIN CENTER1
. DATATRAIN VREF
Final Link Training
. RXDESKEW
. DATATRAIN CENTER2
. LINKSPEED
. REPAIR

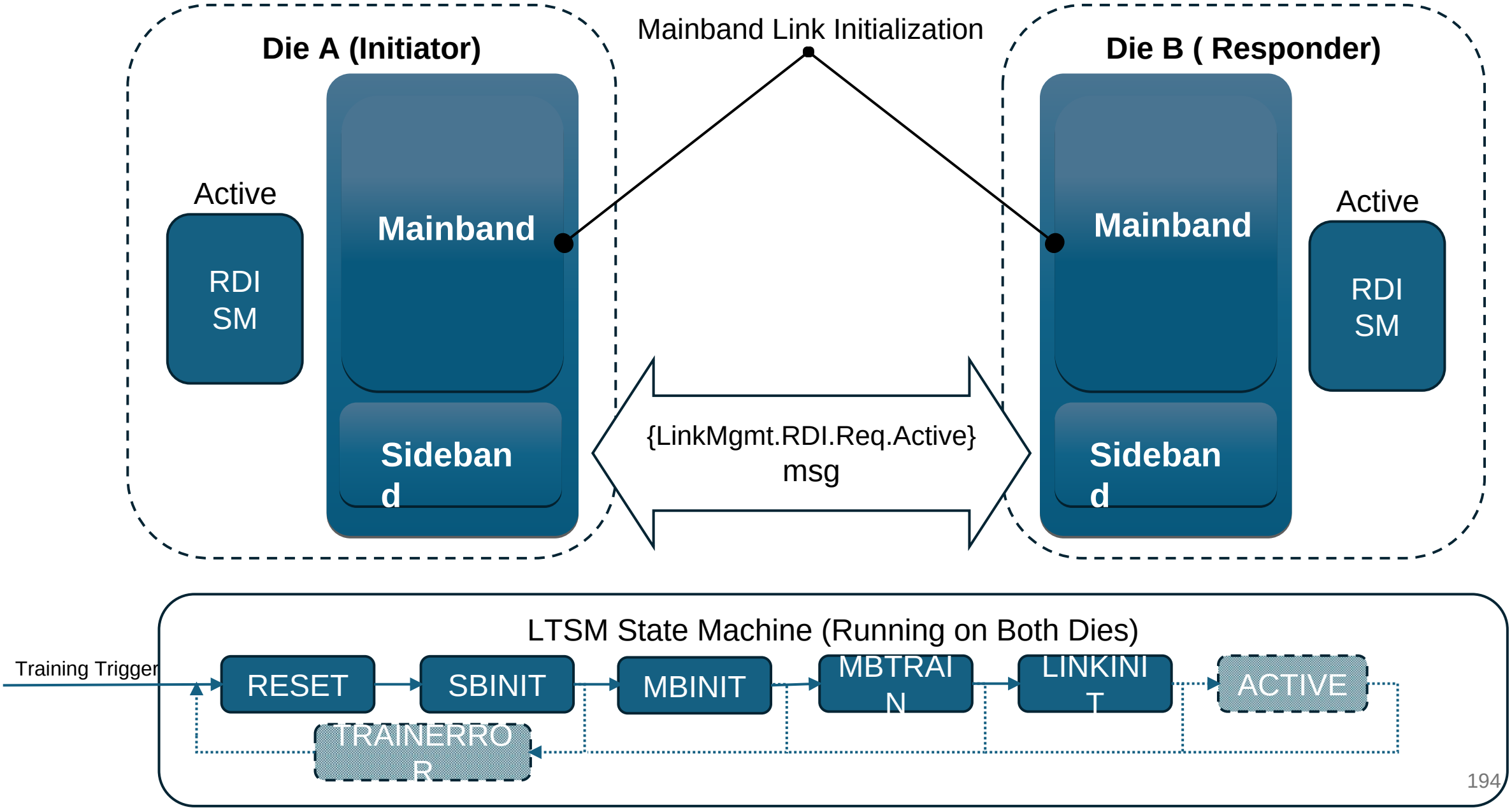
Signal Calibration
Timing Training
Final Link
LINKINI
T

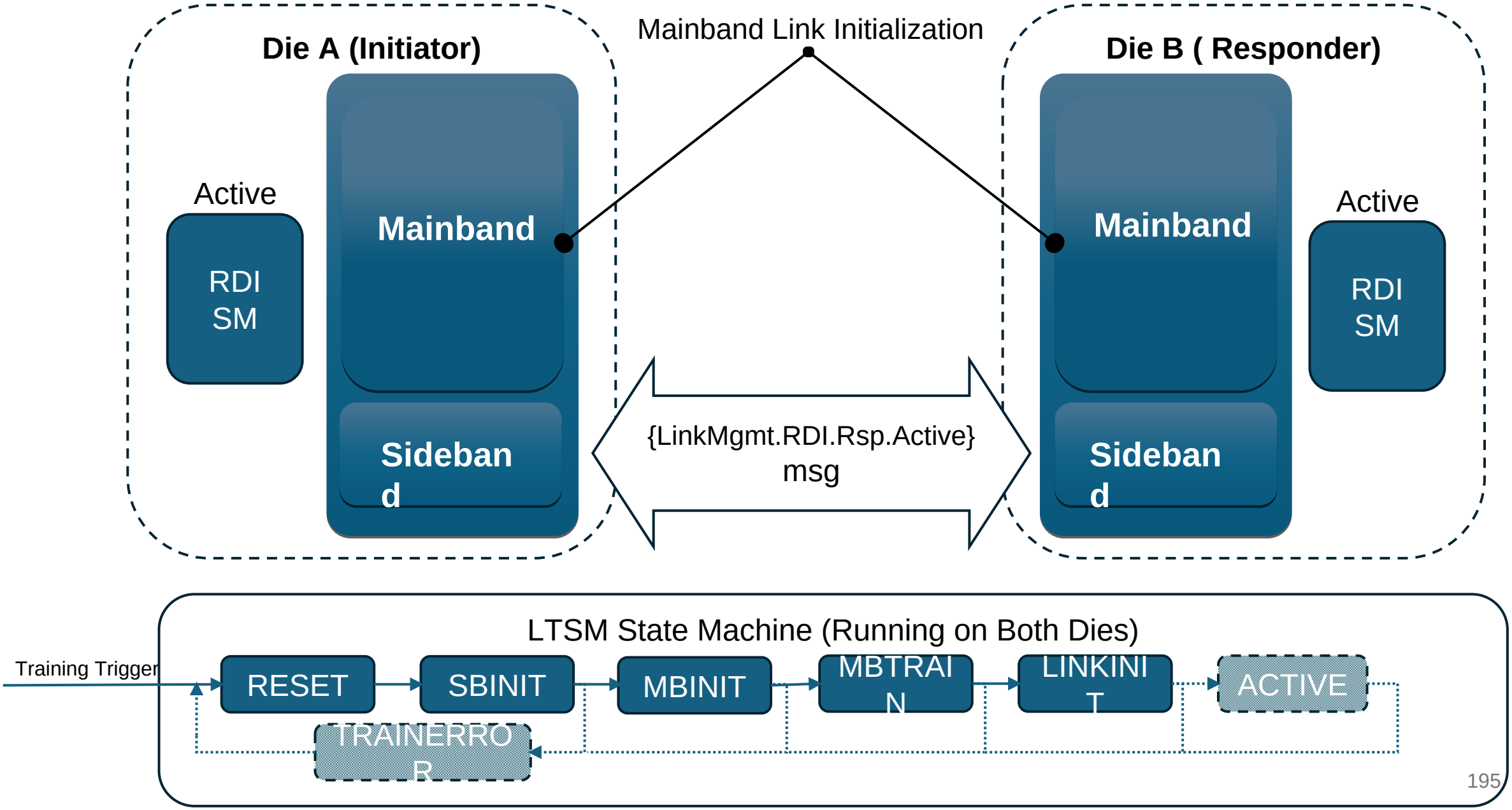
MBTRAIN
Pre-Speed Baseline
. VALVREF
. DATAVREF
. SPEED IDLE
Fine Training (At Target Rate)
. TXSELFCAL
. RXCLKCAL
. VALTRAIN CENTER
. VALTRAIN VREF
. DATATRAIN CENTER1
. DATATRAIN VREF
. RXDESKEW
. DATATRAIN CENTER2
Verification & Repair
. LINKSPEED
. REPAIR

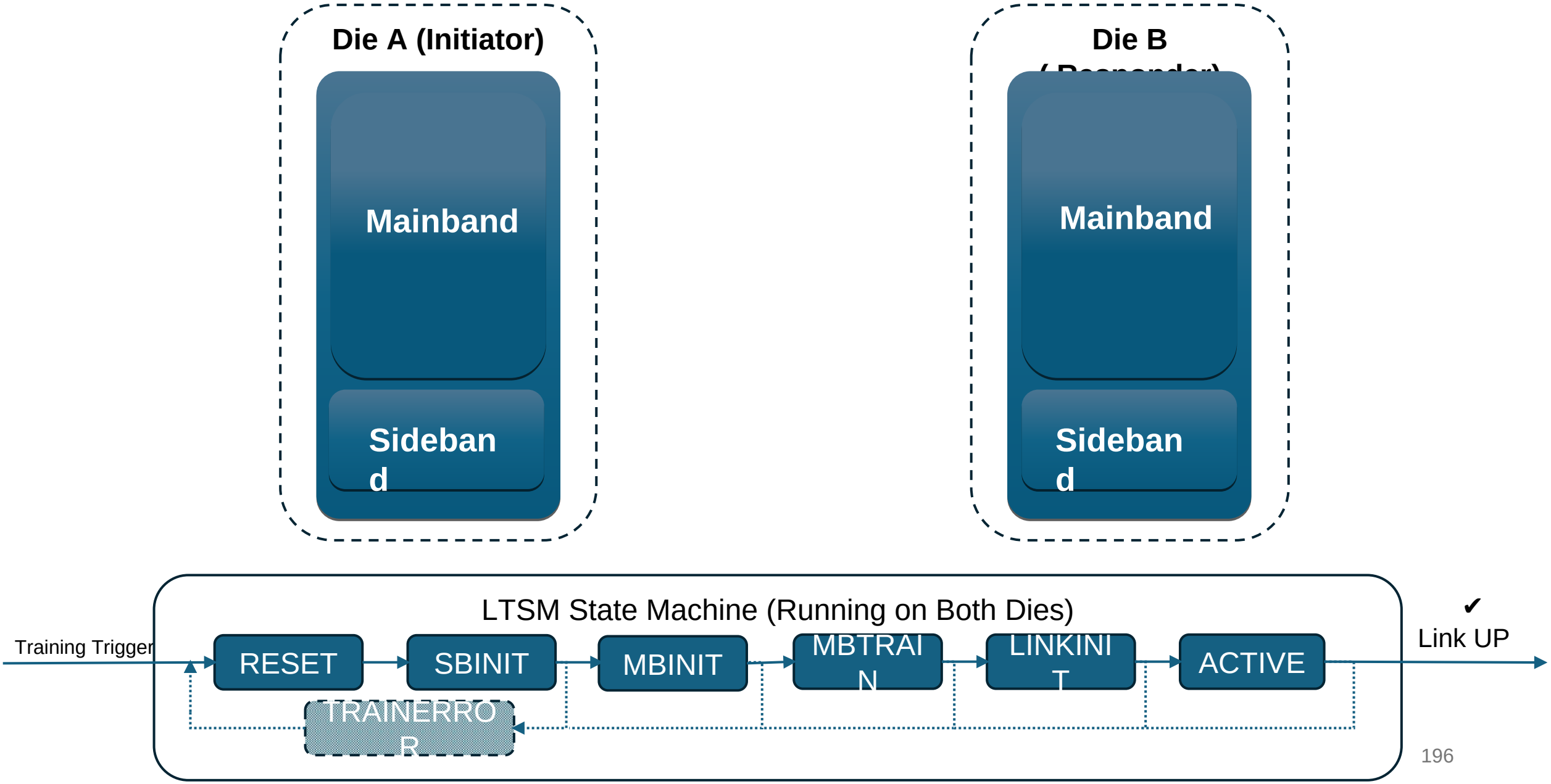
Pre-Speed
Fine Training
Verification &
LINKINI
T

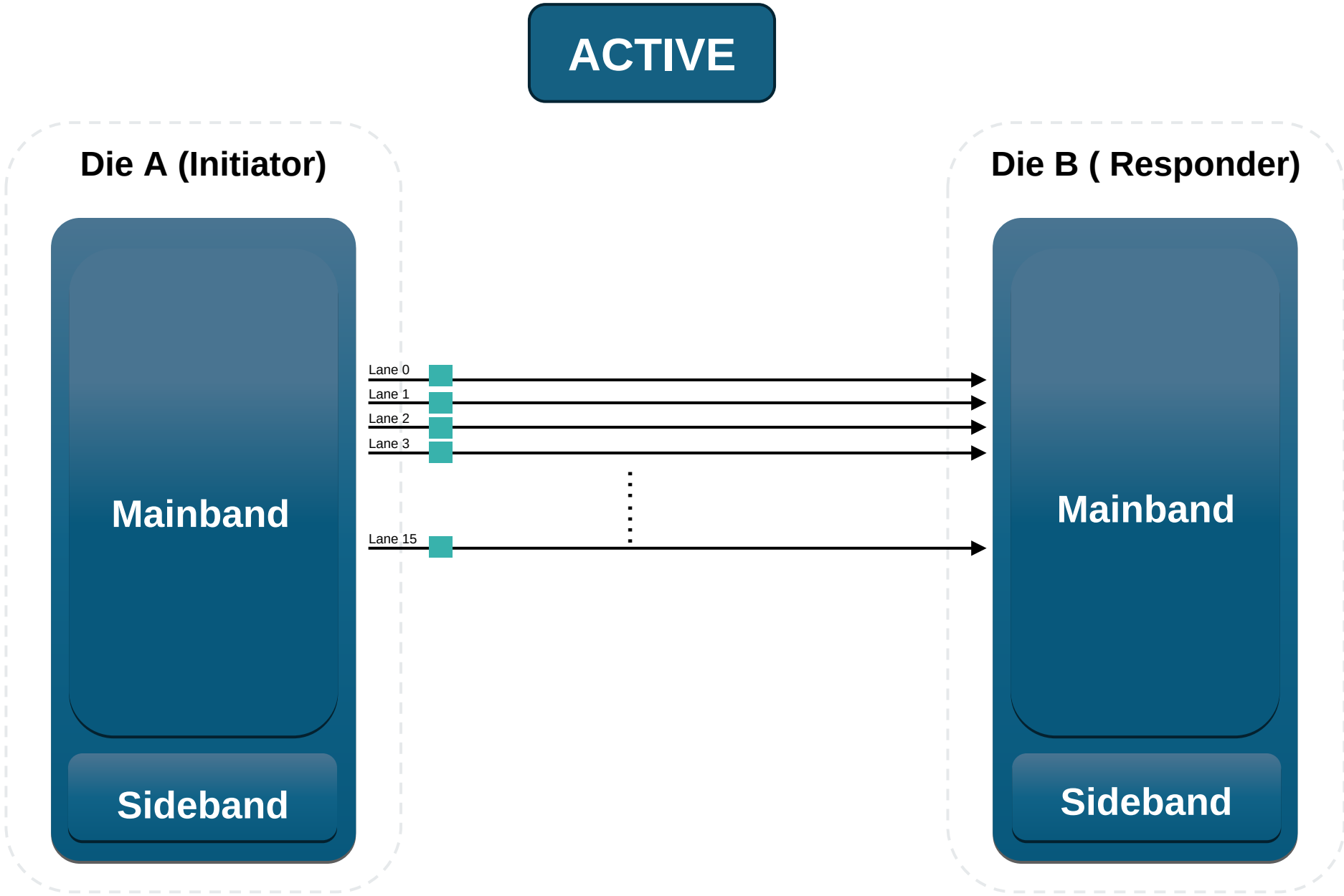

Die A (Initiator)
Mainband Link Initialization
Die B
Mainband
Sideband
Mainband
Sideband

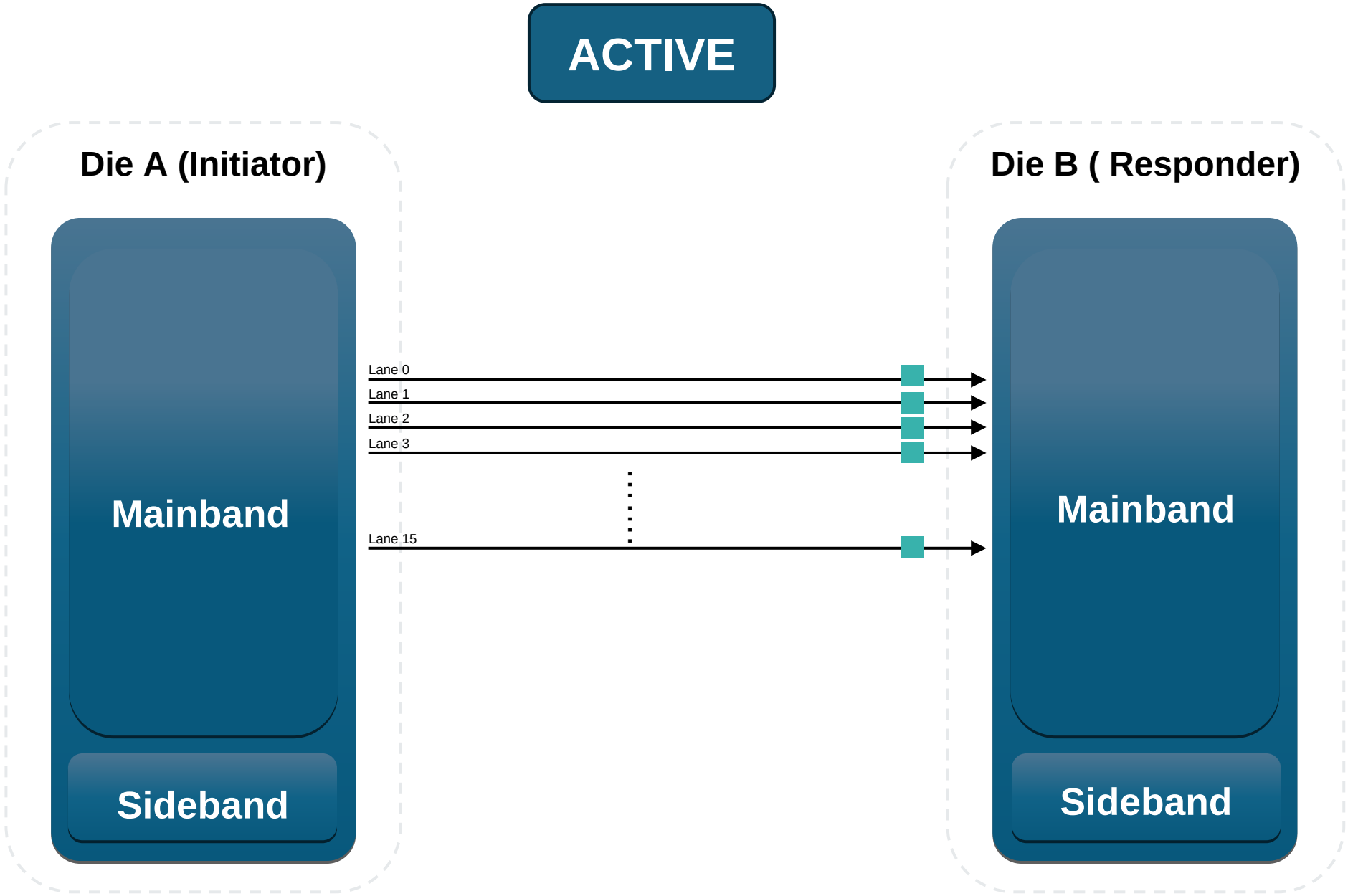

LTSM State Machine (Running on Both Dies)
Training Trigger
RESET
SBINIT
MBINIT
MBTRAIN
LINKINIT
ACTIVE
TRAINERROR


Mainband Link Initialization
Die A (Initiator)
Die B (Responder)
Active
RDI
SM
Mainband
Sideband
Mainband
Sideband
Active
RDI
SM
{LinkMgmt.RDI.Req.Active}
msg
LTSM State Machine (Running on Both Dies)
Training Trigger
RESET
SBINIT
MBINIT
MBTRAIN
LINKINIT
ACTIVE
TRAINERROR


Mainband Link Initialization
Die A (Initiator)
Die B (Responder)
Active
RDI
SM
Mainband
Sideband
Mainband
Sideband
Active
RDI
SM
{LinkMgmt.RDI.Rsp.Active}
msg
LTSM State Machine (Running on Both Dies)
Training Trigger
RESET
SBINIT
MBINIT
MBTRAIN
LINKINIT
ACTIVE
TRAINERROR

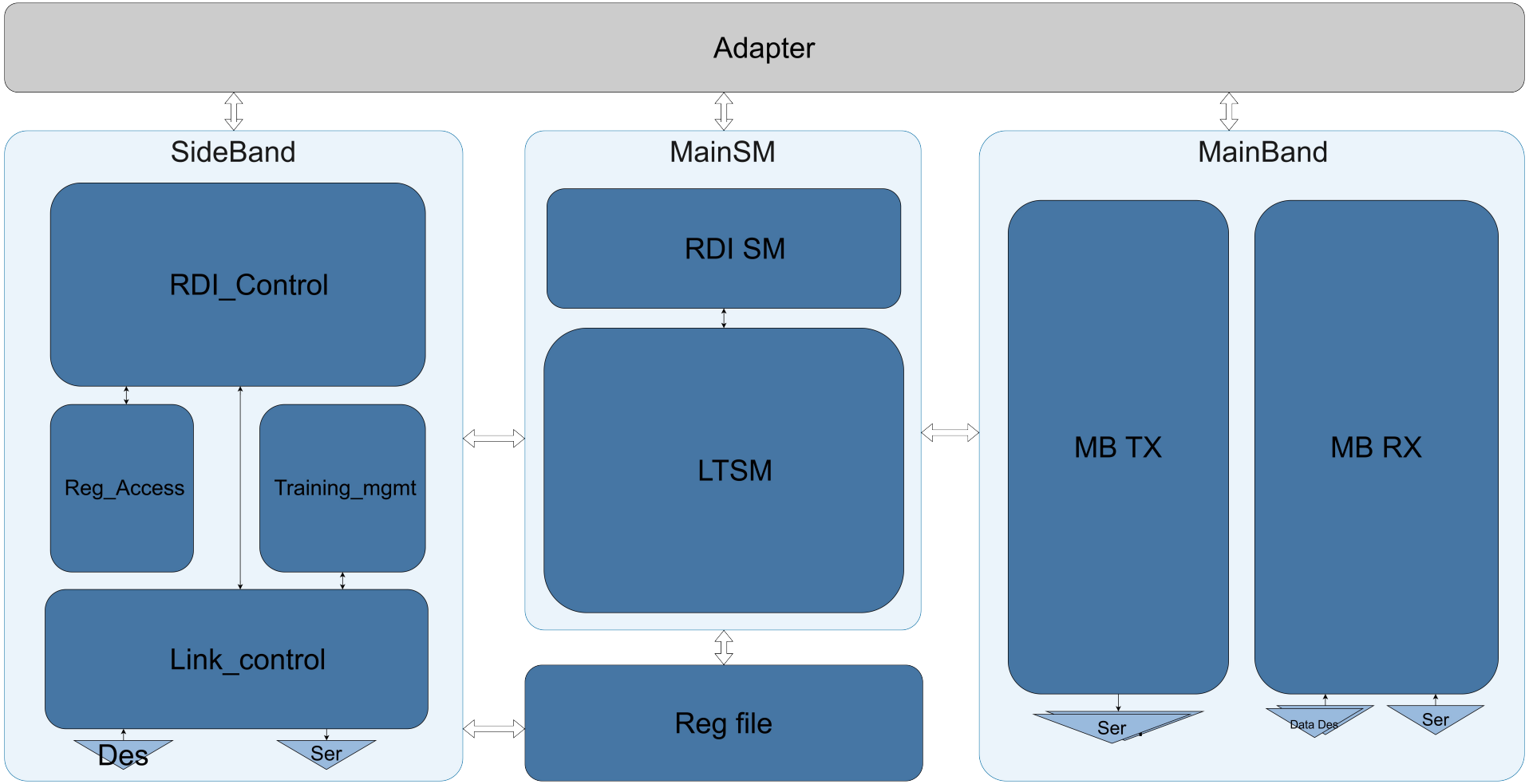

Die A (Initiator)
Mainband
Sideband
Die B
Mainband
Sideband
LTSM State Machine (Running on Both Dies)
Training Trigger
RESET
SBINIT
MBINIT
MBTRAIN
LINKINIT
ACTIVE
TRAINERROR
✔
Link UP


ACTIVE
Die A (Initiator)
Die B (Responder)
Mainband
Sideband
Lane 0
Lane 1
Lane 2
Lane 3
Lane 15
Mainband
Sideband

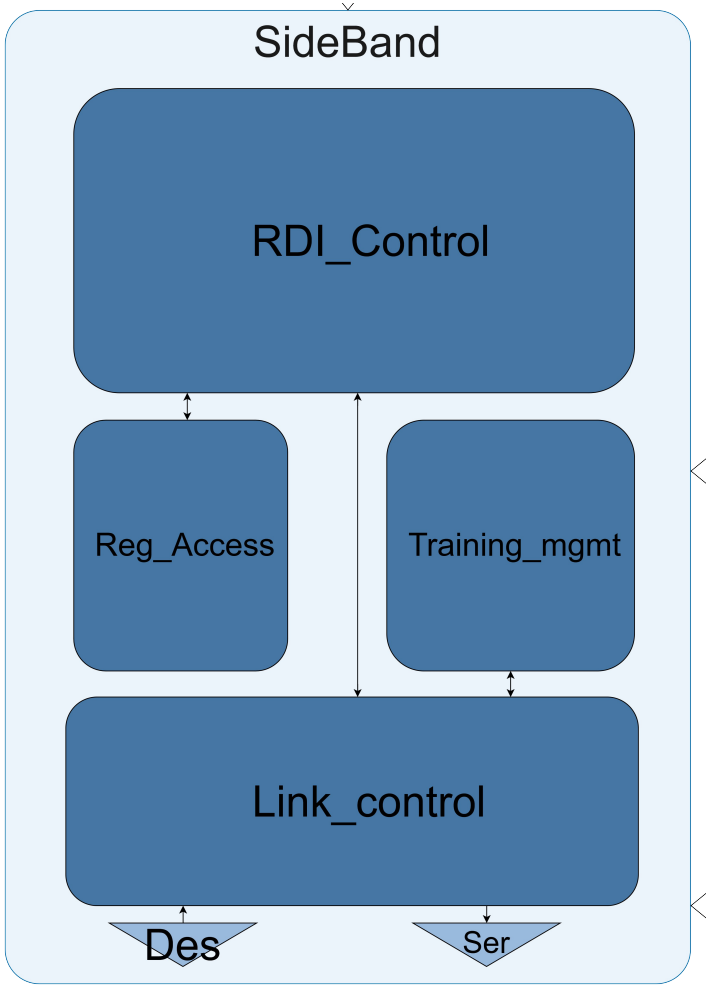

ACTIVE
Die A (Initiator)
Mainband
Sideband
Lane 0
Lane 1
Lane 2
Lane 3
Lane 15
Die B (Responder)
Mainband
Sideband

Verification

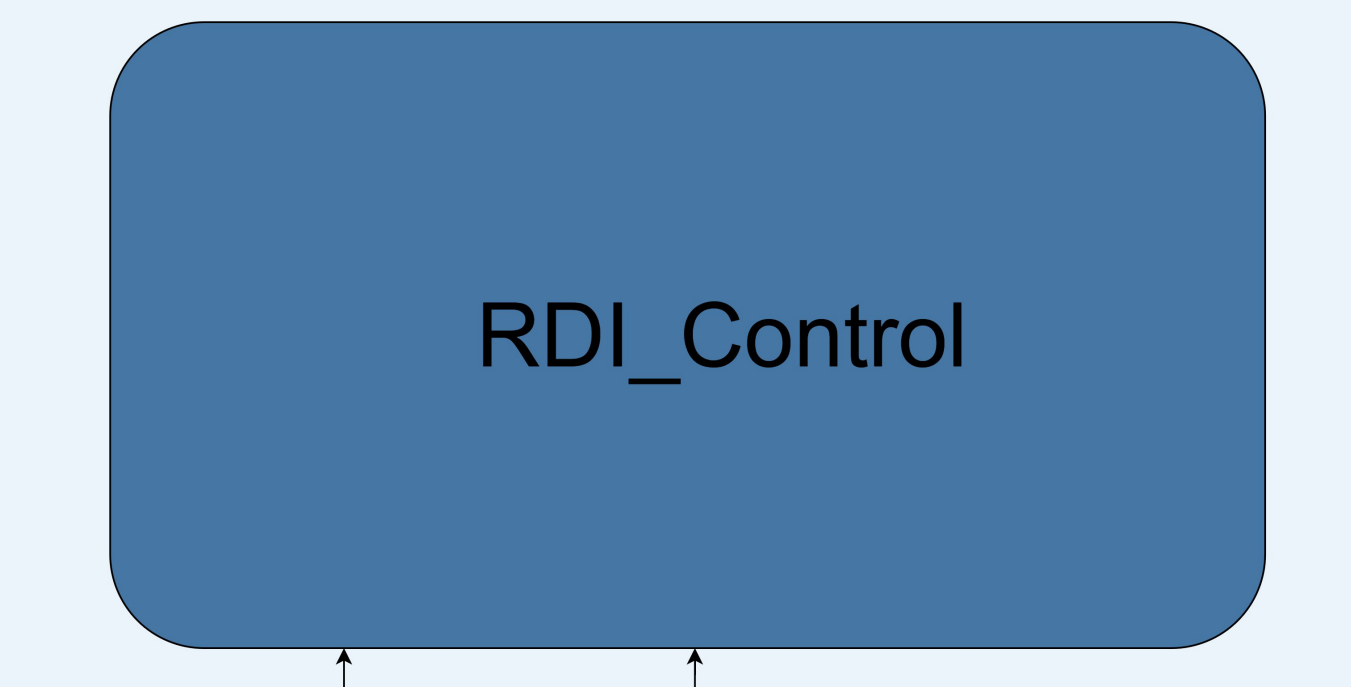
erification Strategies



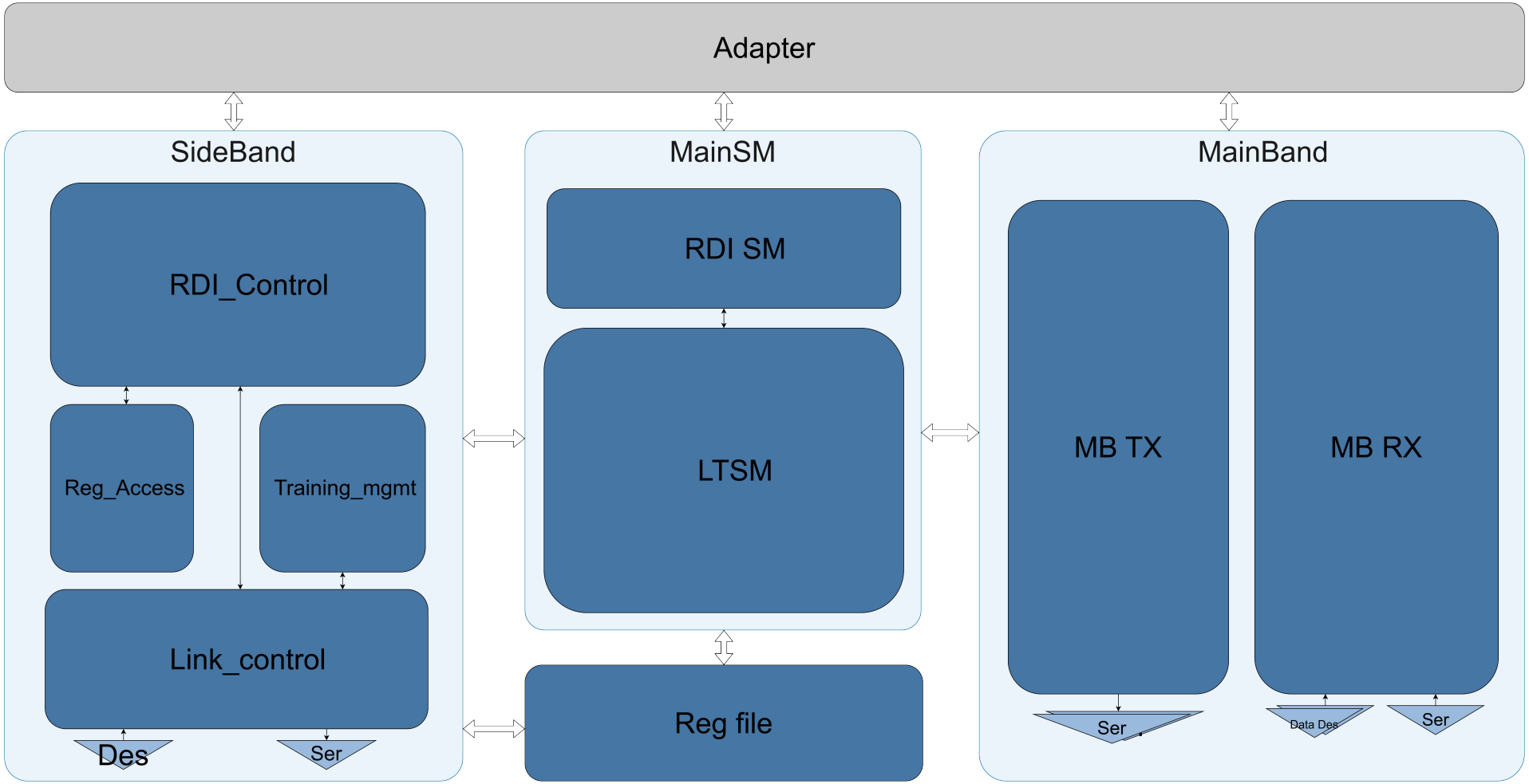
erification Strategies



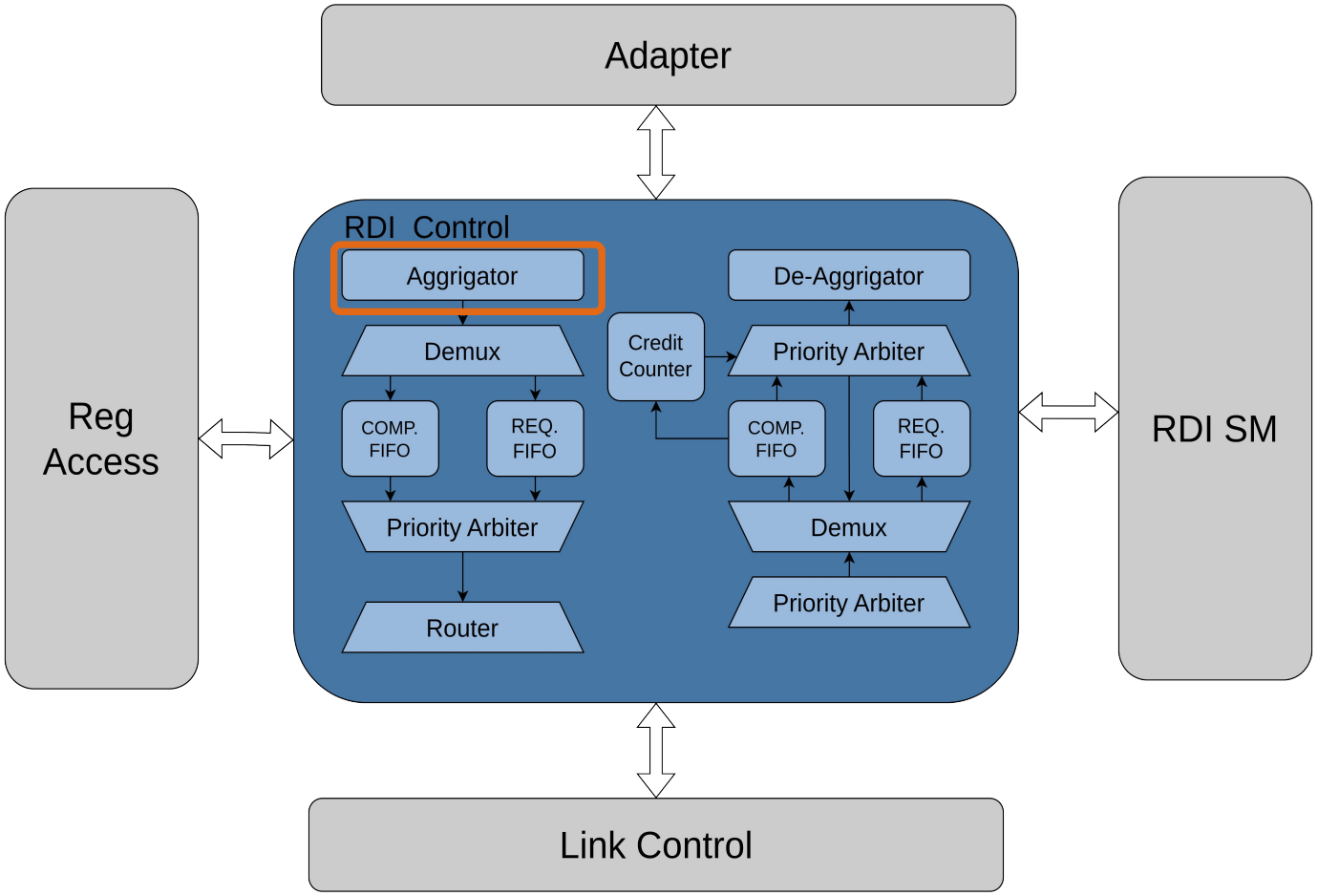
erification Strategies



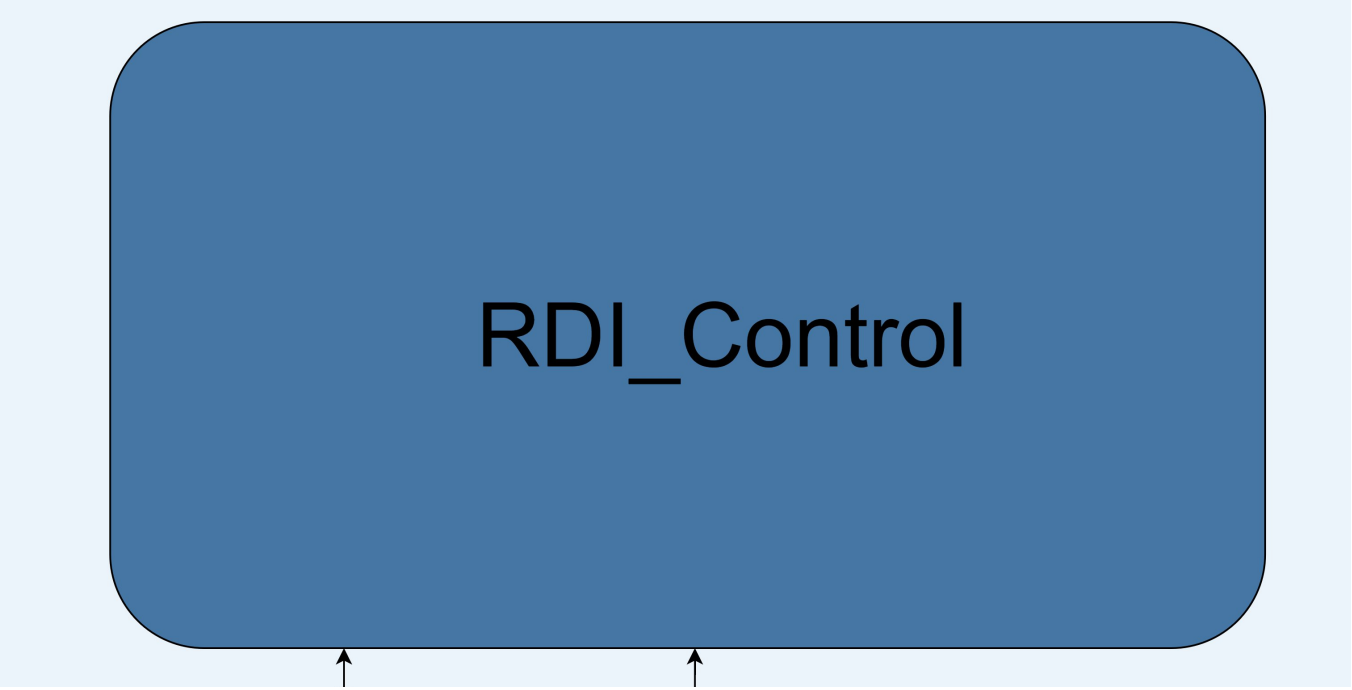
Cle Architecture



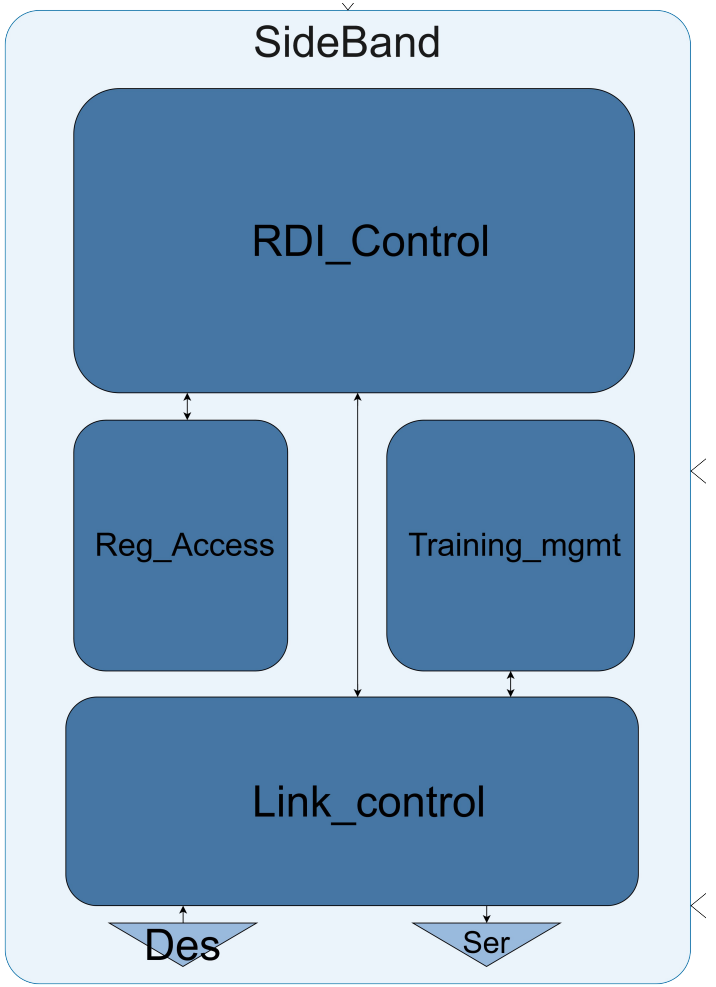
hit Verification



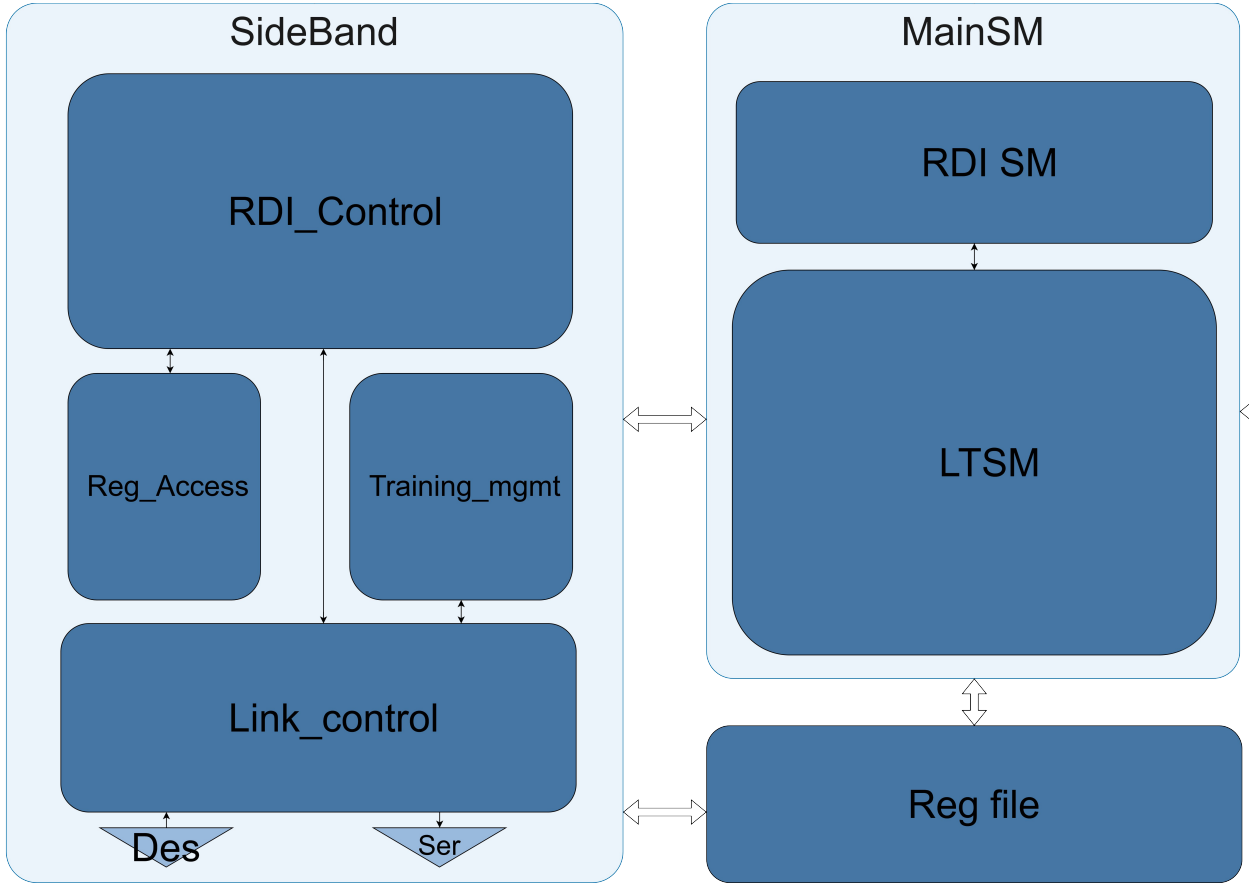
rapper Verification



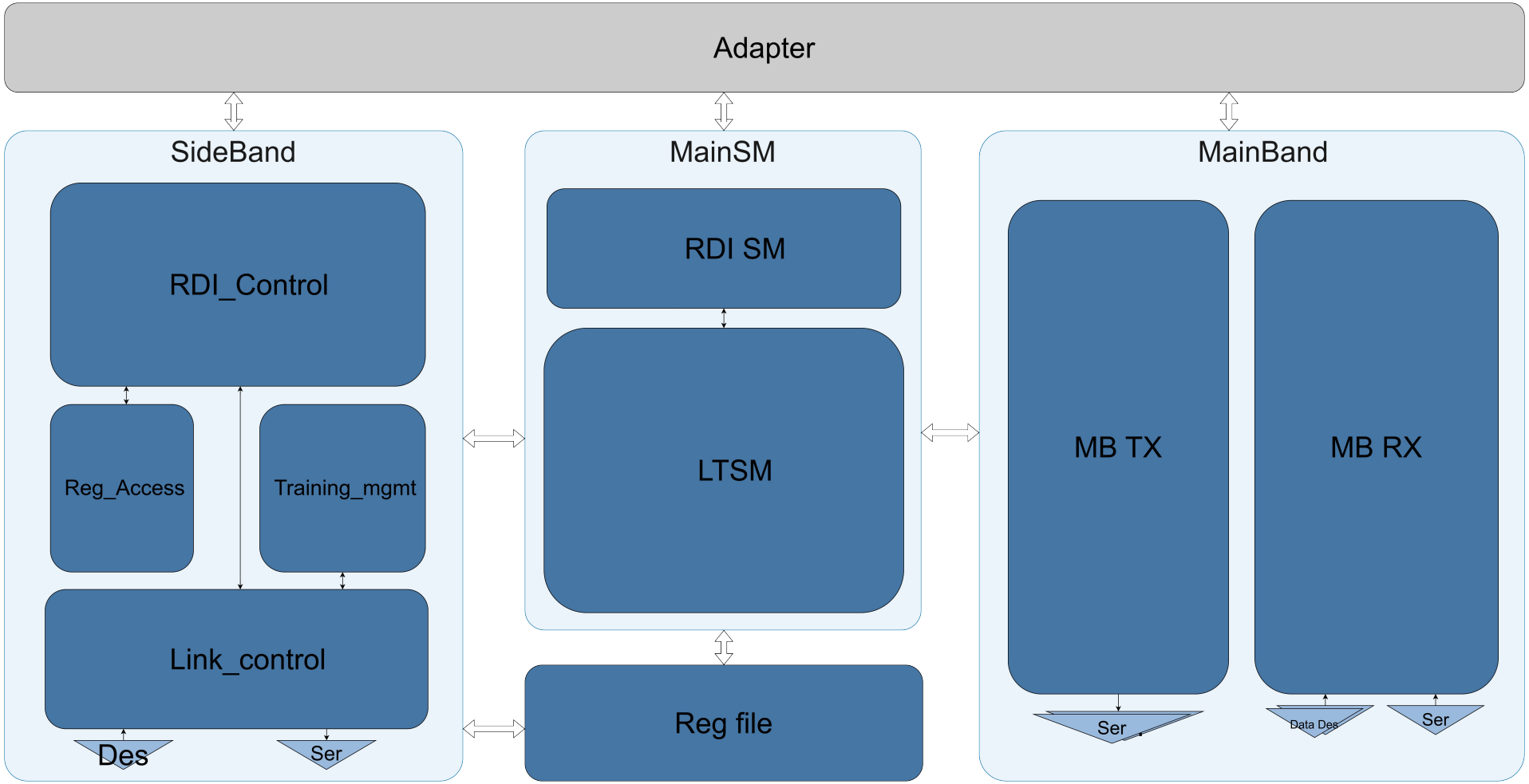
rapper Verification



tegration Verification



ystem Verification



Verification strategy & Environment

1

Unit

Sub-blocks in isolation (e.g. serializer SVA, timer, D2C point-test FSM)

2

Wrapper

Substate wrapper / protocol engine with behavioral stubs for context

3

Integration

Multiple subsystems joined (die-to-die, full MBTRAIN + substates)

4

System

Full UCIe PHY exercised across complete link init & active transfer

Simulation Environment

Mentor QuestaSim (ModelSim) on Linux and Windows. Linux: Makefile with CONFIG / TOP variables (make run or make debug). Windows: run_sim.ps1 equivalent. Listfiles separate source sets from testbench control.

Self-Checking Methodology

Every testbench is autonomous: pass/fail counters track outcomes and report a PASS/FAIL banner. A global watchdog timer flags stalled handshakes. Shared checker attachments (ltsm_tb_if / ltsm_tb_attachments) are reused across all 13 MBTRAIN substate testbenches.

System-Level Verification of the UCIe PHY

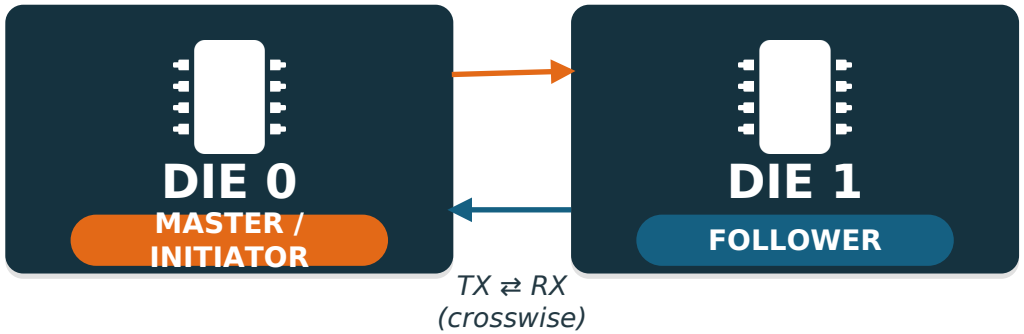
Top of the verification pyramid — proving the PHY as a real two-die chiplet link

WHAT THE SYSTEM TESTBENCH PROVES

- **Two independent dies** — powered up from reset, not a single wrapped block
- **Configured only via registers** — brought up through the sideband register interface
- **Negotiate over real serial wires** — each die sees the other only through physical pins
- **Exchange real data** — full main-band datapath, nothing short-circuited

Bench: *UCIe_PHY_wrapper_tb.sv (~2150 lines) drives the pin-identical UCIe_PHY_wrapper DUT.*

TWO-DIE "SILICON-REALISTIC" TOPOLOGY



MAIN-BAND + SIDEBAND CROSS-CONNECT

16 data lanes | valid | fwd clock P/N | track lane | 2-wire sideband

A true link bring-up — not a loopback

How the Link Is Exercised & Checked

Real config path, fault-injecting channel, adapter-side stimulus, and one self-checking verdict

Fault-Injecting Channel

Cross-connect is a behavioral channel, not a plain wire.
Stuck-at → lane repair / width degrade
Bit-reversal → lane-reversal detect & correct
Valid-lane flip → PHYRETRAIN
Block sideband → watchdog / TRAINERROR
DUT is unaware — must recover via spec training only.

Real Register Path

Nothing force-programmed — sideband transactions stream into each die's Reg_File.
3 registers / die: Training Setup 4, PHY Control, UCIe Link Control (+ start-training on die-0).
Config / decode path verified end-to-end.

RDI (Adapter) Face

Bench plays the adapter above each PHY.
Bring-up sequencing to LOG_ACTIVE.
Clock- and stall-handshake responders.
PM & error stimuli: L1 / L2 entry, wake, link-error.

Checking & Data Proof

Single scoreboard chk() → one PASS / FAIL verdict across the whole 29-scenario regression.
Full-duplex proof: DEADBEEF ⇄ CAFEBABE confirms the entire datapath end-to-end.
White-box probes on LTSM & RDI states.

Coverage of Intent — 29 System Scenarios

A coverage grid over UCIe’s four training axes, plus power and error management

29

SCENARIOS

Individually gated by enabled_scenarios[1..29]; run as one PASS/FAIL regression via make run under QuestaSim.

1 Bring-up & negotiation
SC 1, 3, 9, 16

2 Lane reversal
SC 4, 5

3 Width degradation / repair
SC 6–8, 14

4 Speed degradation
SC 15, 20, 21, 27

5 Combined-axis stress
SC 17–19, 22–26

6 Power management
SC 10 (L1) · 11 (L2→RESET)

7 Runtime error & recovery
SC 12, 13, 28, 2

8 Sideband integrity
SC 29

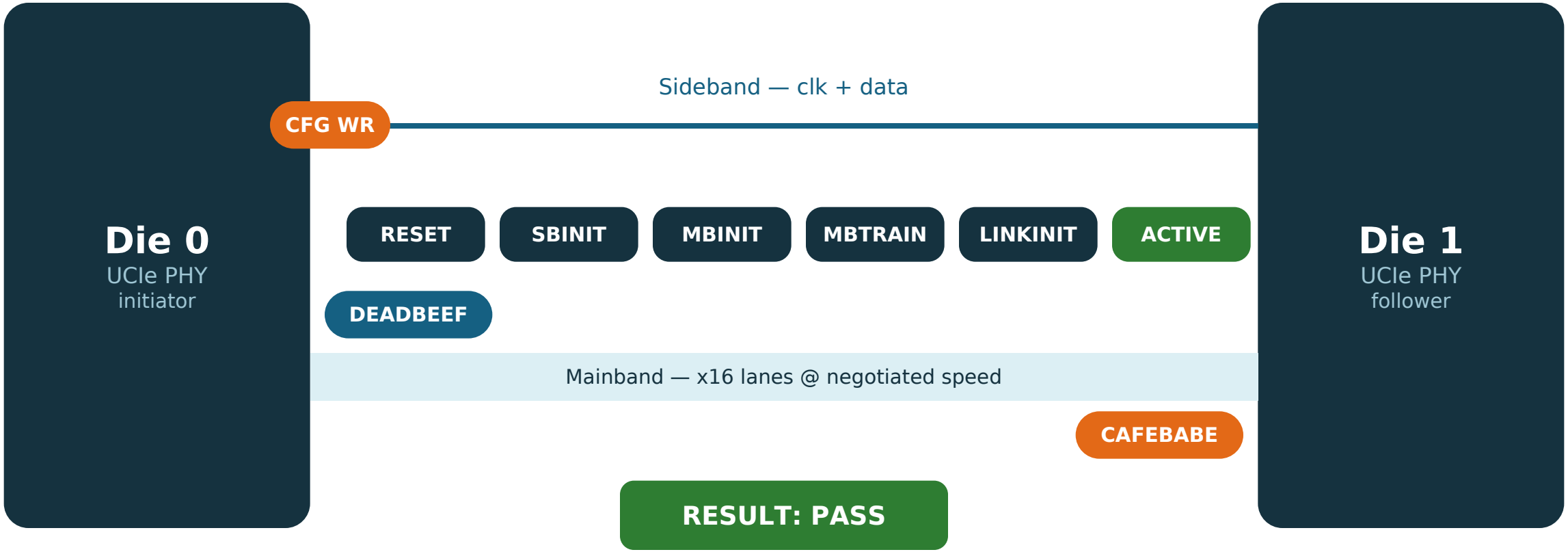
Not ad-hoc: the 29 scenarios span every meaningful combination of reversal × width × speed, plus power and runtime-error recovery — the hardest cases for the negotiation logic.

anscript snippet

```
# T=44451000 | [DIE0] ltsm_n=LOG_MBTRAIN_TXSELFCAL rdi_sts=Reset pl_state_sts=Reset
# T=44491000 | [DIE1] ltsm_n=LOG_MBTRAIN_TXSELFCAL rdi_sts=Reset pl_state_sts=Reset
# T=44811000 | [DIE0] ltsm_n=LOG_MBTRAIN_RXCLKCAL rdi_sts=Reset pl_state_sts=Reset
# T=44851000 | [DIE1] ltsm_n=LOG_MBTRAIN_RXCLKCAL rdi_sts=Reset pl_state_sts=Reset
# T=45501000 | [DIE0] ltsm_n=LOG_MBTRAIN_VALTRAINCENTER rdi_sts=Reset pl_state_sts=Reset
# T=45541000 | [DIE1] ltsm_n=LOG_MBTRAIN_VALTRAINCENTER rdi_sts=Reset pl_state_sts=Reset
# T=51071000 | [DIE0] ltsm_n=LOG_MBTRAIN_VALTRAINVREF rdi_sts=Reset pl_state_sts=Reset
# T=51111000 | [DIE1] ltsm_n=LOG_MBTRAIN_VALTRAINVREF rdi_sts=Reset pl_state_sts=Reset
# T=55981000 | [DIE0] ltsm_n=LOG_MBTRAIN_DATATRAINCENTER1 rdi_sts=Reset pl_state_sts=Reset
# T=56021000 | [DIE1] ltsm_n=LOG_MBTRAIN_DATATRAINCENTER1 rdi_sts=Reset pl_state_sts=Reset
# T=61851000 | [DIE0] ltsm_n=LOG_MBTRAIN_DATATRAINVREF rdi_sts=Reset pl_state_sts=Reset
# T=61891000 | [DIE1] ltsm_n=LOG_MBTRAIN_DATATRAINVREF rdi_sts=Reset pl_state_sts=Reset
# T=67061000 | [DIE0] ltsm_n=LOG_MBTRAIN_RXDESKEW rdi_sts=Reset pl_state_sts=Reset
# T=67101000 | [DIE1] ltsm_n=LOG_MBTRAIN_RXDESKEW rdi_sts=Reset pl_state_sts=Reset
# T=72271000 | [DIE0] ltsm_n=LOG_MBTRAIN_DATATRAINCENTER2 rdi_sts=Reset pl_state_sts=Reset
# T=72311000 | [DIE1] ltsm_n=LOG_MBTRAIN_DATATRAINCENTER2 rdi_sts=Reset pl_state_sts=Reset
# T=78140000 | [DIE0] ltsm_n=LOG_MBTRAIN_LINKSPEED rdi_sts=Reset pl_state_sts=Reset
# T=78180000 | [DIE1] ltsm_n=LOG_MBTRAIN_LINKSPEED rdi_sts=Reset pl_state_sts=Reset
# T=80552000 | [DIE0] ltsm_n=LOG_LINKINIT rdi_sts=Reset pl_state_sts=Reset
# T=80592000 | [DIE1] ltsm_n=LOG_LINKINIT rdi_sts=Reset pl_state_sts=Reset
# T=80895000 | [DIE0] ltsm_n=LOG_LINKINIT rdi_sts=Active pl_state_sts=Reset
# T=80896000 | [DIE0] ltsm_n=LOG_ACTIVE rdi_sts=Active pl_state_sts=Reset
# T=80900000 | [DIE0] ltsm_n=LOG_ACTIVE rdi_sts=Active pl_state_sts=Active
# T=80935000 | [DIE1] ltsm_n=LOG_LINKINIT rdi_sts=Active pl_state_sts=Reset
# T=80936000 | [DIE1] ltsm_n=LOG_ACTIVE rdi_sts=Active pl_state_sts=Reset
# T=80940000 | [DIE1] ltsm_n=LOG_ACTIVE rdi_sts=Active pl_state_sts=Active
# T=80940000 | [SC1] [PASS]: both dies reached LOG_ACTIVE
#
# +================================================================+
# |                  ACTIVE LINK STATUS REPORT                     |
# +================================================================+
#   [DIE 0] state=LOG_ACTIVE, RDI_sts=Active
#           Width :     x16 (code: 2)
#           Speed : Gen6  (32 GT/s) (code: 5)
#           TX Lane Mask: 3'b011, RX Lane Mask: 3'b011
#           Active Deser Lanes (RX): 16'hffff
#           Lane Reversal: 0  Width Degrade: 0
#   ----------------------------------------------------------------
#   [DIE 1] state=LOG_ACTIVE, RDI_sts=Active
#           Width :     x16 (code: 2)
#           Speed : Gen6  (32 GT/s) (code: 5)
#           TX Lane Mask: 3'b011, RX Lane Mask: 3'b011
#           Active Deser Lanes (RX): 16'hffff
#           Lane Reversal: 0  Width Degrade: 0
# +================================================================+
#
# T=80968000 | [DATA] Die0 received CAFEBABE correctly.
# T=80968000 | [DATA] Die1 received DEADBEEF correctly.
# T=80968000 | [SC1] [PASS]: data transfer verified successfully
```

Scenario: Happy Path Bring-Up

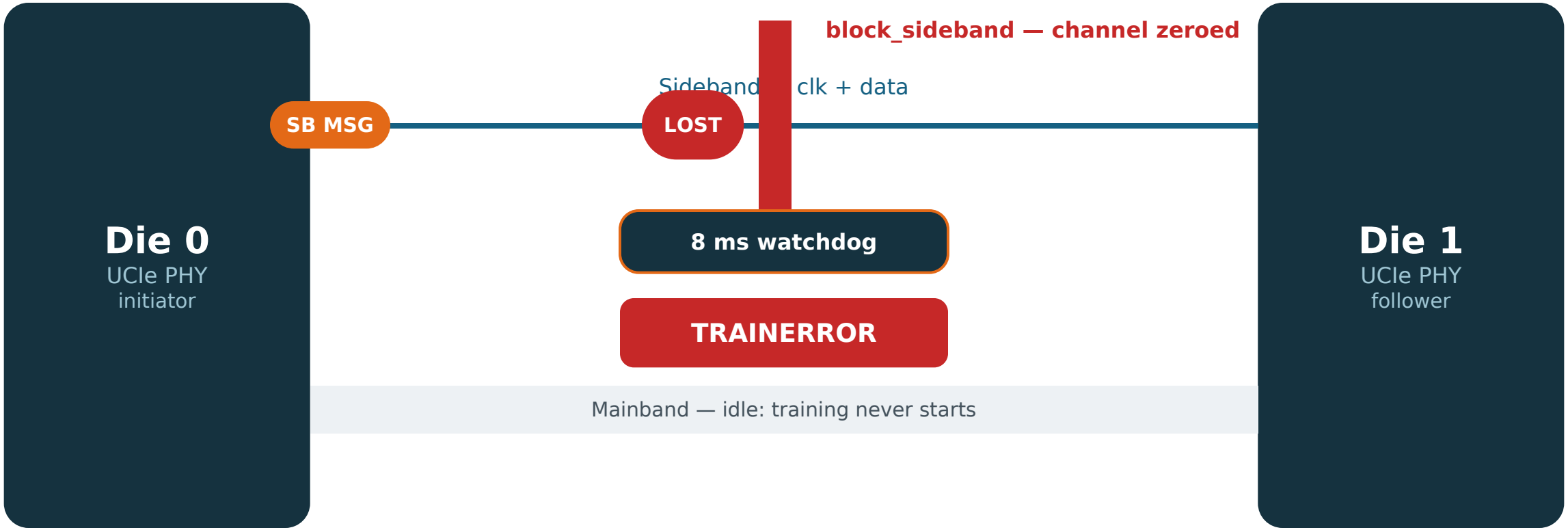
SC 1



3 — Adapter programs protocol IDs, enables Streaming; Start UCIe Link-Training

Scenario: Sideband Blocked → Watchdog

SC 2



1 — Both dies enter SBINIT and begin sideband handshake.

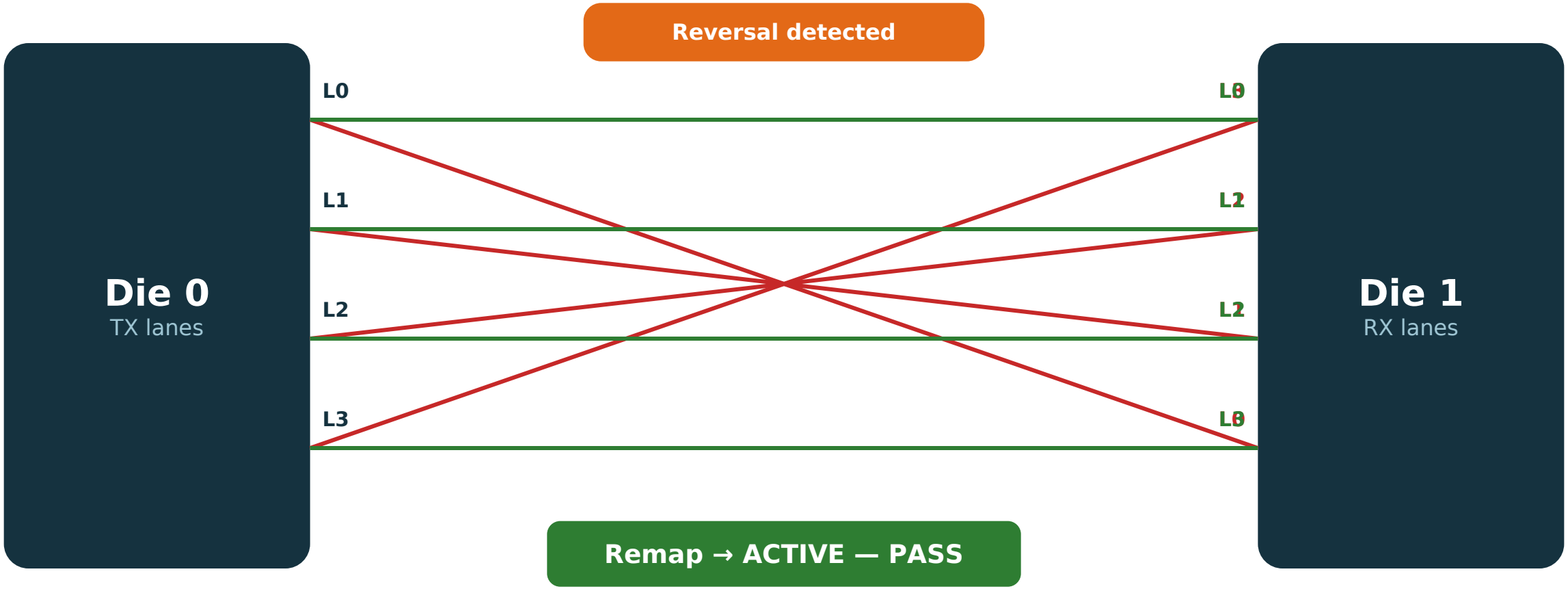
2 — The sideband channel is zeroed; no packets reach the remote die in either direction.

3 — Each die's 8 ms watchdog expires → LTSM enters TRAINERROR.

4 — TRAINERROR is broadcast over the sideband (which is blocked), so the remote die never learns of the error.

Scenario: Lane Reversal

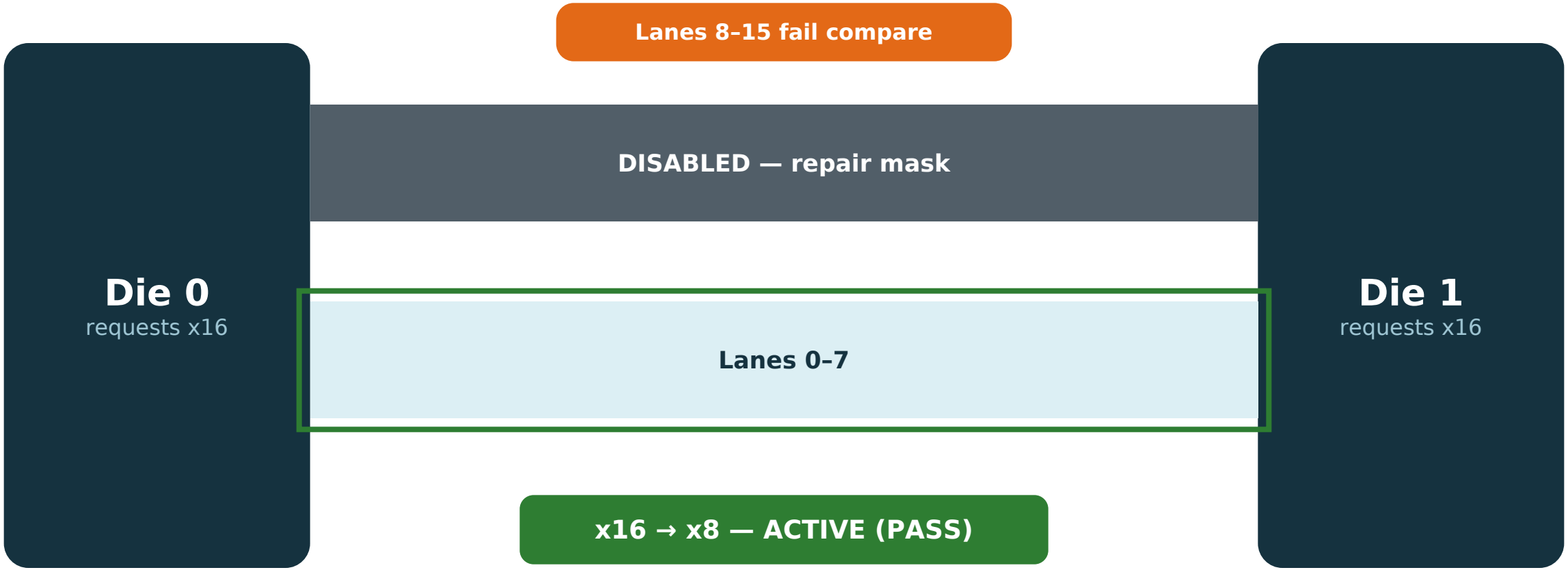
SC 4



3 — TS1 sequences on reversed lanes → detect reversal → remap RX lanes in reverse order

Scenario: Width Degradation

SC 6



2 — The link degrades to x8 and keeps operating on the surviving lanes

Verification Results Summary

Mainband
SerDes + TX→RX loopbacks · framed / stream paths · die-to-die 5×5×3 fault sweep · full training sequence

Sideband
SVA formal on SER / DES (4 + 6 properties) · PMO behavioral: back-to-back + backward compatibility

LTSM / D2C
D2C unit + wrapper (100 randomized iterations) · all 13 MBTRAIN substates · top-level integration · OOP class-based suite

RDI SM
Dual timer (1 µs / 16 ms) · 9 per-state directed TBs · stall handshake + clock gating

Register File
Attribute enforcement (RO / HWInit / RW / RW1C / ROS) · address decode + UR · 32 / 64-bit + byte-enable

Overall Status
Every testbench across all three subsystems — Completed & Verified

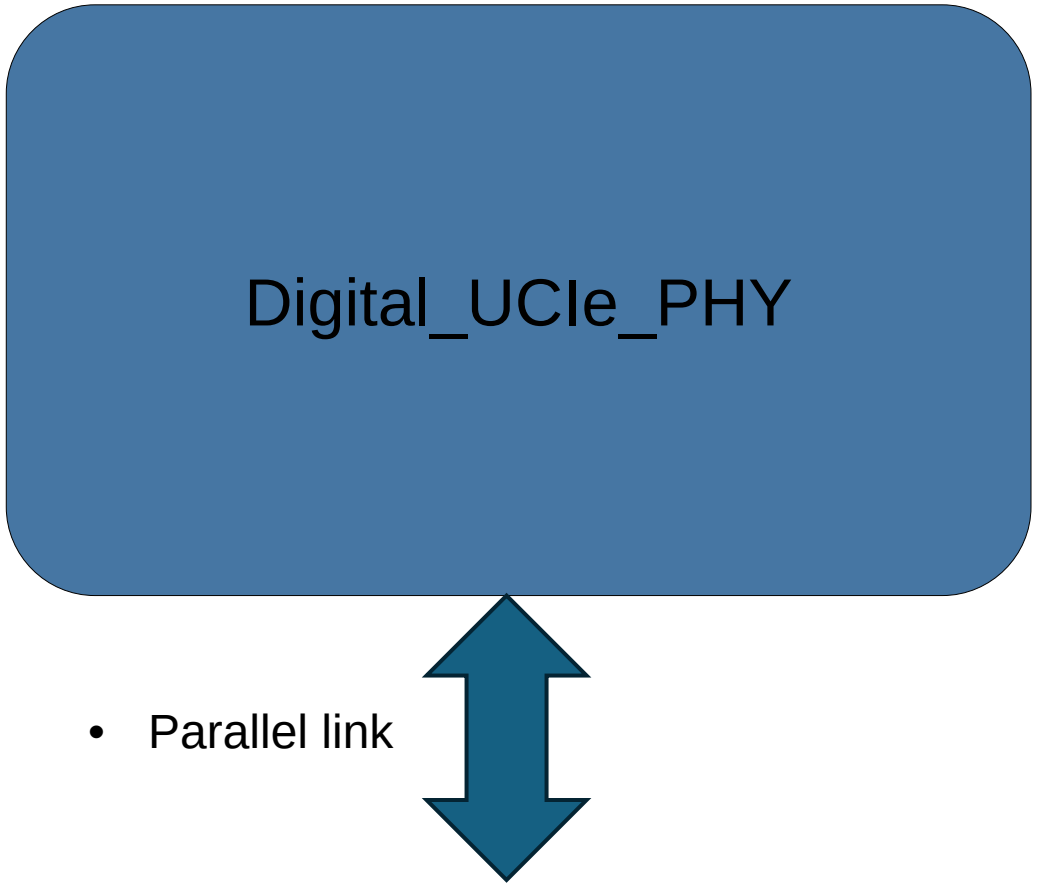
System-level regression: 29 scenarios · one self-checking verdict — **RESULT: PASS** (0 failing checks)

Implementation

FPGA Prototype

Code ready for sythesis

- External clock source
- Serializer & deserializer
- PLL’s
- Clock divider
- Clock gate



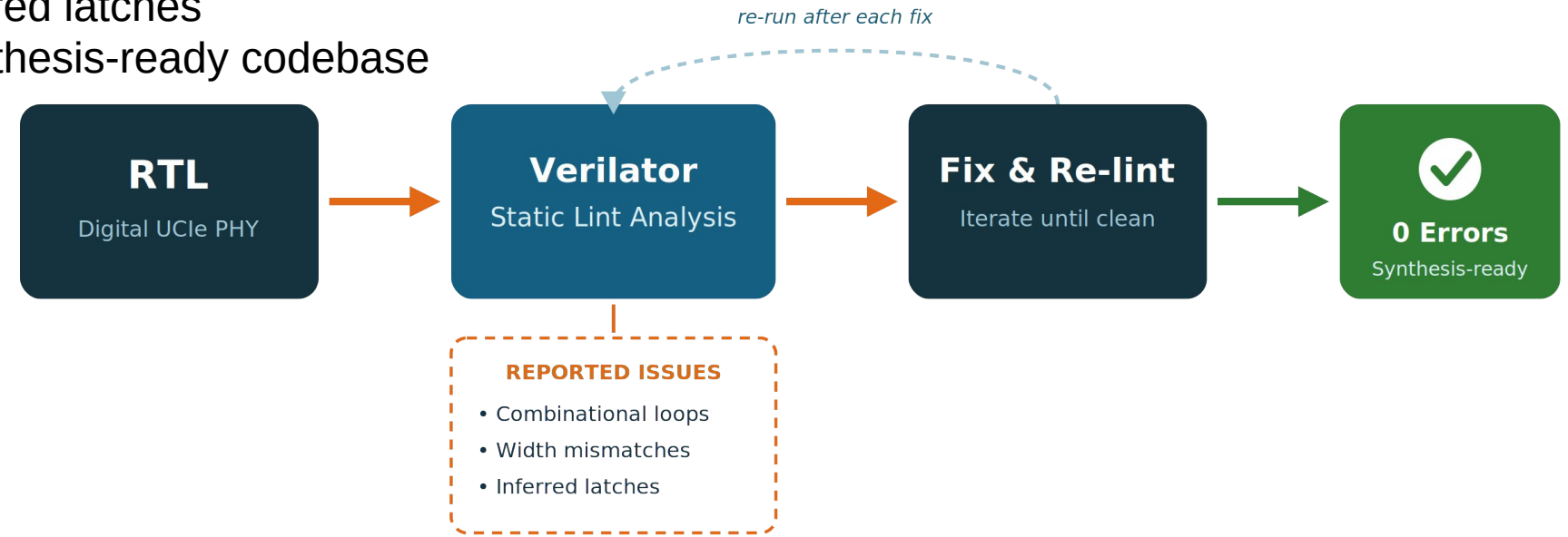
- Parallel link

FPGA Prototype

Code ready for synthesis

Lint error free ✓

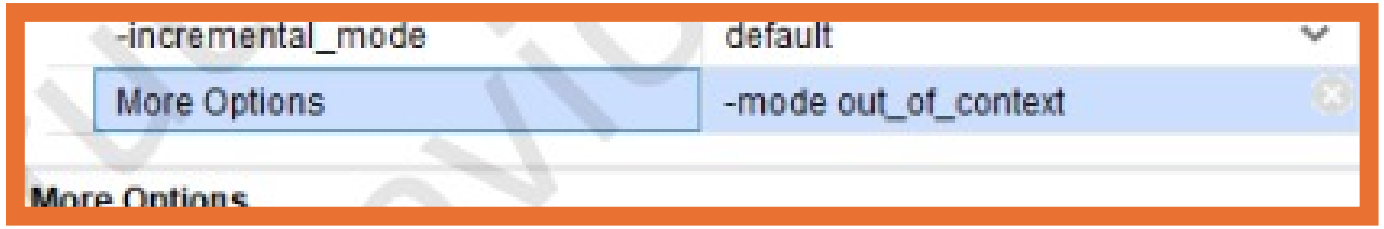
- Full RTL linted with Verilator before synthesis
- Critical fixes: combinational loops, width mismatches, inferred latches
- Result: clean, synthesis-ready codebase



FPGA Prototype

Synthesis

- Target: Zynq UltraScale+ MPSoC (ZU3EG), speed grade -1
- Out-of-context synthesis mode : evaluates PHY timing standalone, without I/O constraints
- Met timing @ **125 MHz**



FPGA Prototype

Synthesis optimization

125

MHz

Max synthesized frequency for the complete digital PHY (all 4 subsystems + UCIe_PHY wrapper). No timing violations at this frequency.

- Explored multiple synthesis + implementation strategies
- Best timing with: **Flow_PerfOptimized_high** synthesis strategy with **Performance_Retiming** implementation strategy

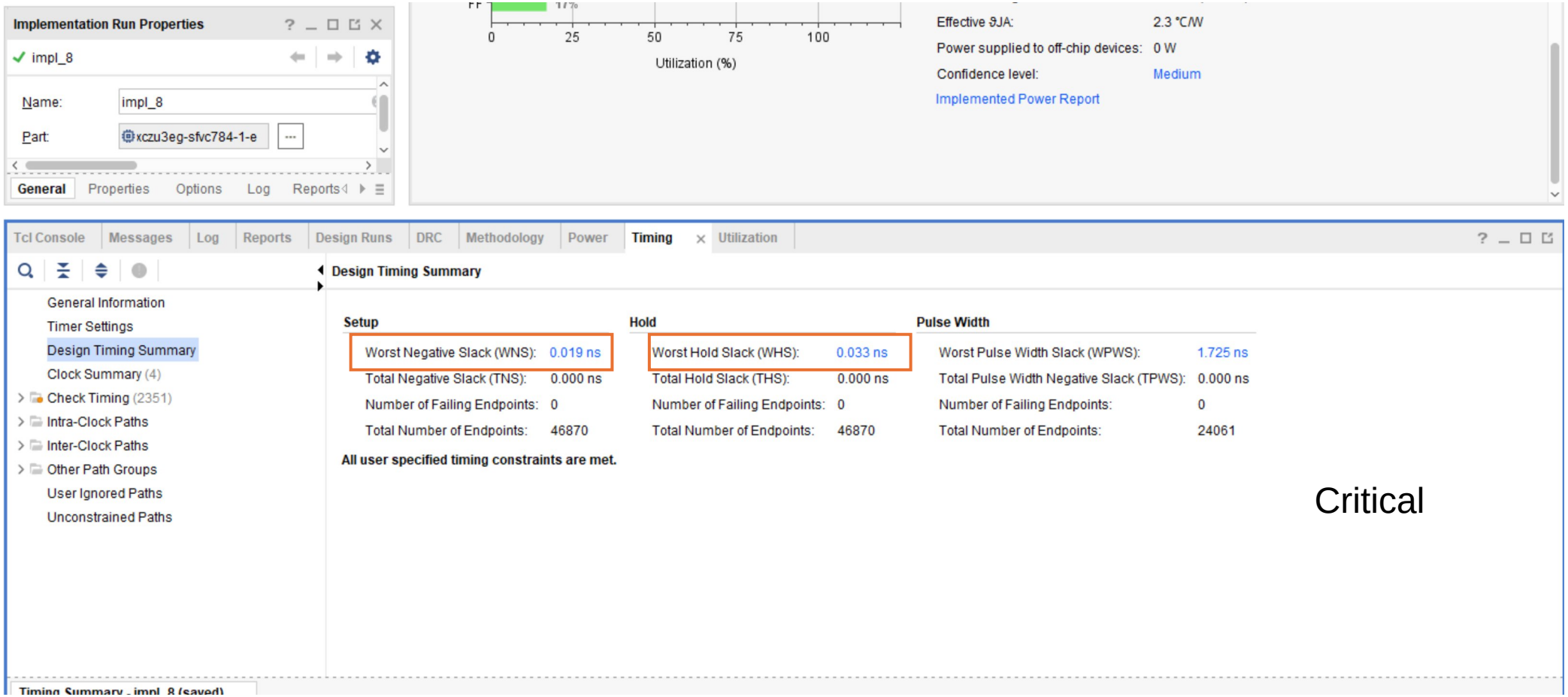
250

MHz

Max synthesized frequency for the complete digital PHY (all 4 subsystems + UCIe_PHY wrapper). No timing violations at this frequency.

FPGA Prototype

Timing report summary


Implementation Run Properties
impl_8
Name: impl_8
Part: xczu3eg-sfvc784-1-e
General Properties Options Log Reports
0 25 50 75 100
Utilization (%)
Effective ϑJA: 2.3 °C/W
Power supplied to off-chip devices: 0 W
Confidence level: Medium
Implemented Power Report
Tcl Console Messages Log Reports Design Runs DRC Methodology Power Timing Utilization
General Information
Timer Settings
Design Timing Summary
Clock Summary (4)
Check Timing (2351)
Intra-Clock Paths
Inter-Clock Paths
Other Path Groups
User Ignored Paths
Unconstrained Paths
Design Timing Summary
Setup
Worst Negative Slack (WNS): 0.019 ns
Total Negative Slack (TNS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 46870
Hold
Worst Hold Slack (WHS): 0.033 ns
Total Hold Slack (THS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 46870
Pulse Width
Worst Pulse Width Slack (WPWS): 1.725 ns
Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0
Total Number of Endpoints: 24061
All user specified timing constraints are met.
Timing Summary - impl_8 (saved)

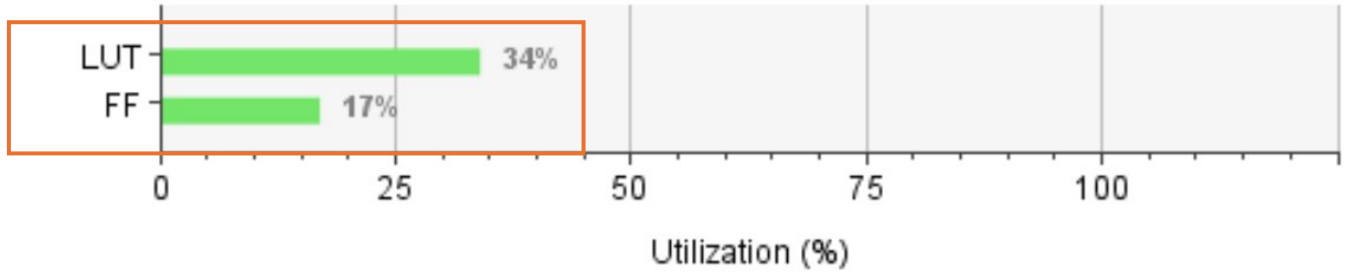
Critical

FPGA Prototype

Utilization report summary

Summary

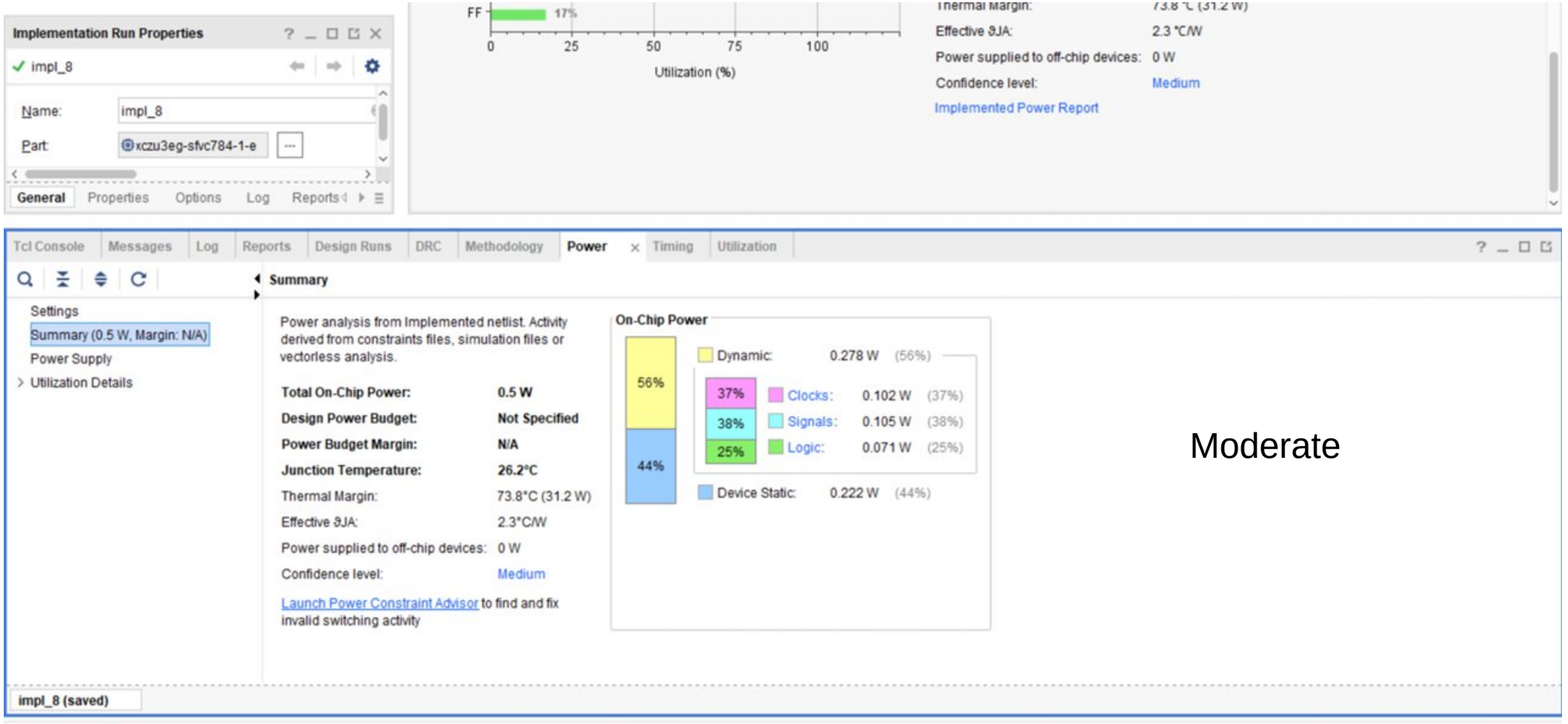
Resource	Utilization	Available	Utilization %
LUT	23956	70560	33.95
FF	24061	141120	17.05



Moderate

FPGA Prototype

Power report summary


Implementation Run Properties
impl_8
Name: impl_8
Part: xczu3eg-sfvc784-1-e
General Properties Options Log Reports
FF 17%
0 25 50 75 100
Utilization (%)
Thermal Margin: 73.8 °C (31.2 W)
Effective ϑJA: 2.3 °C/W
Power supplied to off-chip devices: 0 W
Confidence level: Medium
Implemented Power Report
Tcl Console Messages Log Reports Design Runs DRC Methodology Power Timing Utilization
Summary
Settings
Summary (0.5 W, Margin: N/A)
Power Supply
Utilization Details
Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.
Total On-Chip Power: 0.5 W
Design Power Budget: Not Specified
Power Budget Margin: N/A
Junction Temperature: 26.2°C
Thermal Margin: 73.8°C (31.2 W)
Effective ϑJA: 2.3°C/W
Power supplied to off-chip devices: 0 W
Confidence level: Medium
Launch Power Constraint Advisor to find and fix invalid switching activity
On-Chip Power
56%
44%
Dynamic: 0.278 W (56%)
37% Clocks: 0.102 W (37%)
38% Signals: 0.105 W (38%)
25% Logic: 0.071 W (25%)
Device Static: 0.222 W (44%)
impl_8 (saved)

Moderate

Hardware Validation

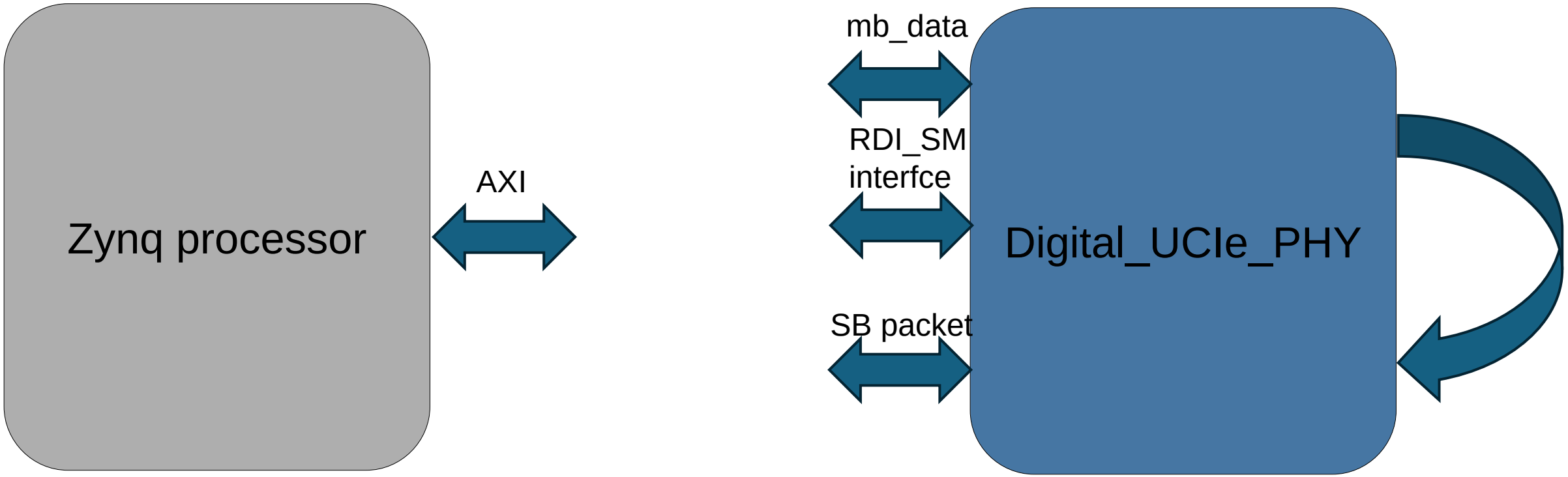
Hardware Validation

Platform & Approach — Avnet Ultra96-V2 (Zynq UltraScale+ MPSoC, ZU3EG)

Analog hard macros (PLL, SerDes, tri-state drivers) stripped; the entire digital PHY kept, with every TX boundary folded back onto its RX as **self-loopback.**

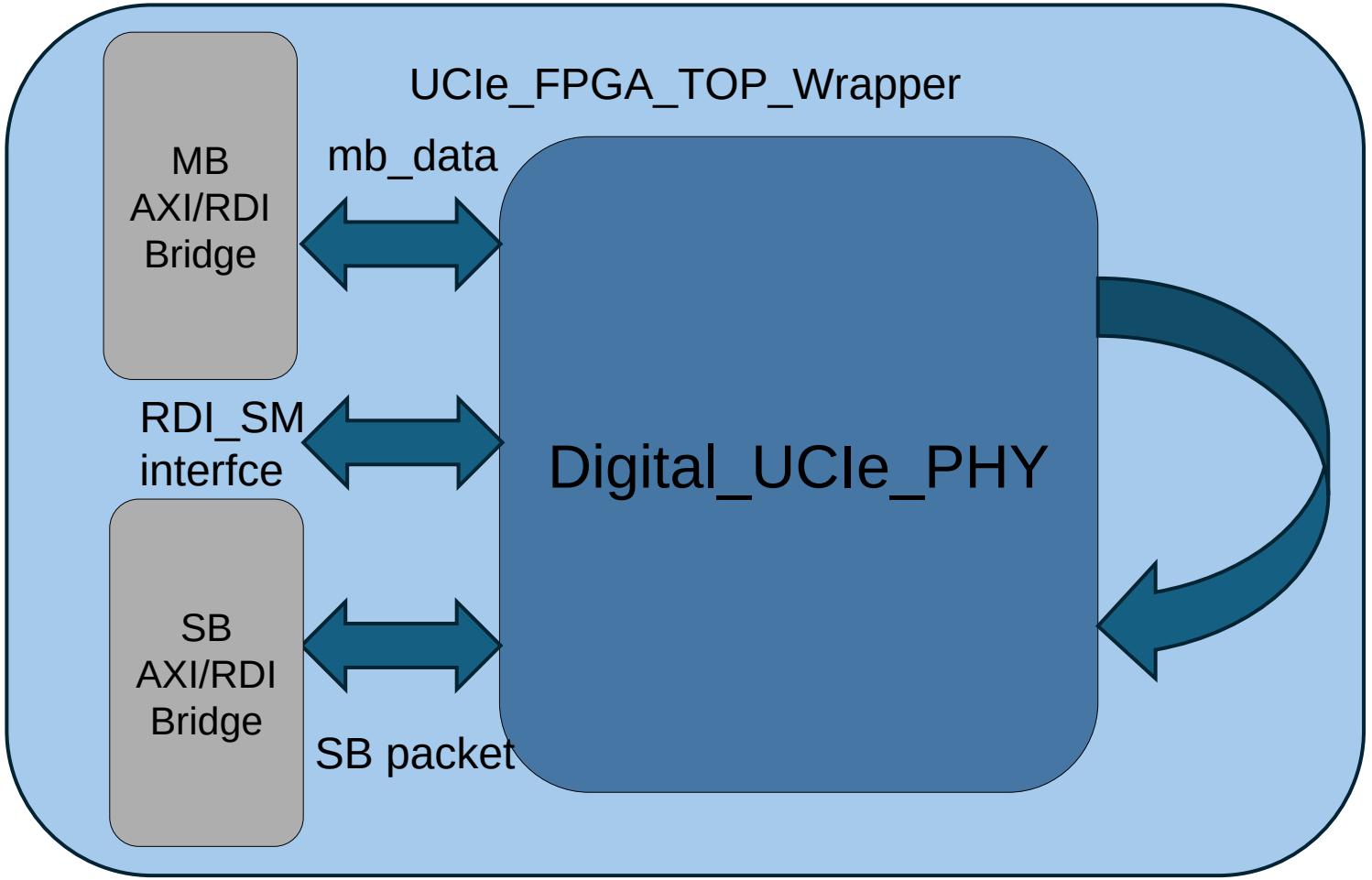
Hardware Validation

Platform & Approach — Avnet Ultra96-V2 (Zynq UltraScale+ MPSoC, ZU3EG)
Analog hard macros (PLL, SerDes, tri-state drivers) stripped; the entire digital PHY kept, with every TX boundary folded back onto its RX as **self-loopback.**

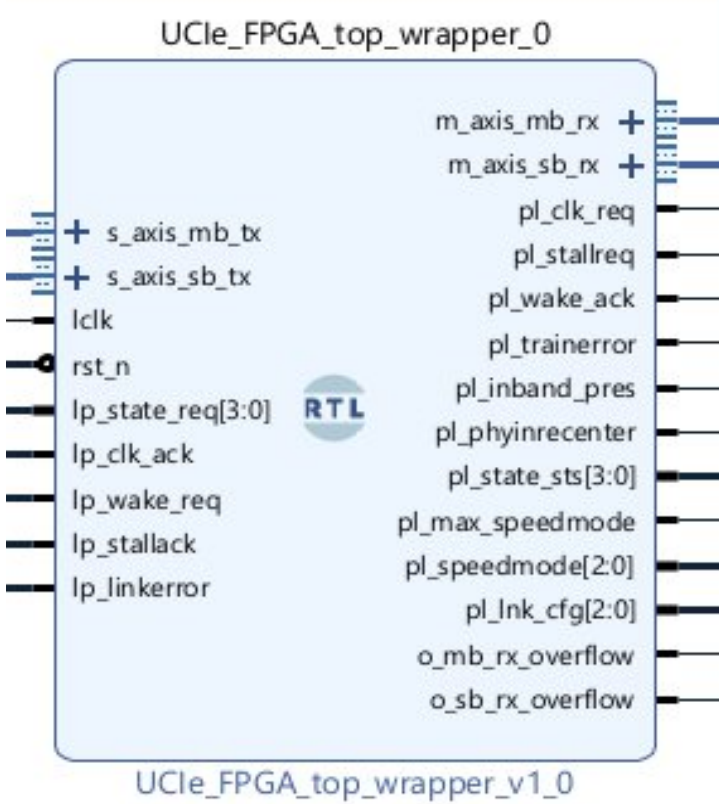


Hardware Validation

- **MB AXI/RDI Bridge** — maps mainband flit data to AXI-Stream (DMA path)
- **SB AXI/RDI Bridge** — maps sideband packets to AXI-Stream (FIFO path)
- **RDI state machine interface** — driven directly via GPIO (EMIO): state requests out, link status in
- PHY becomes a standard software-controlled peripheral

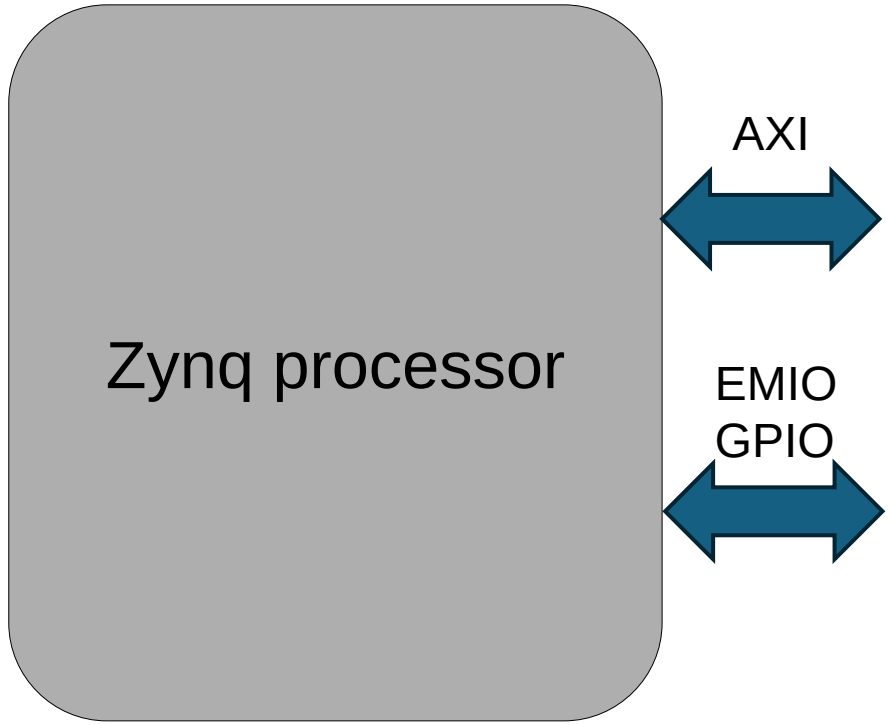


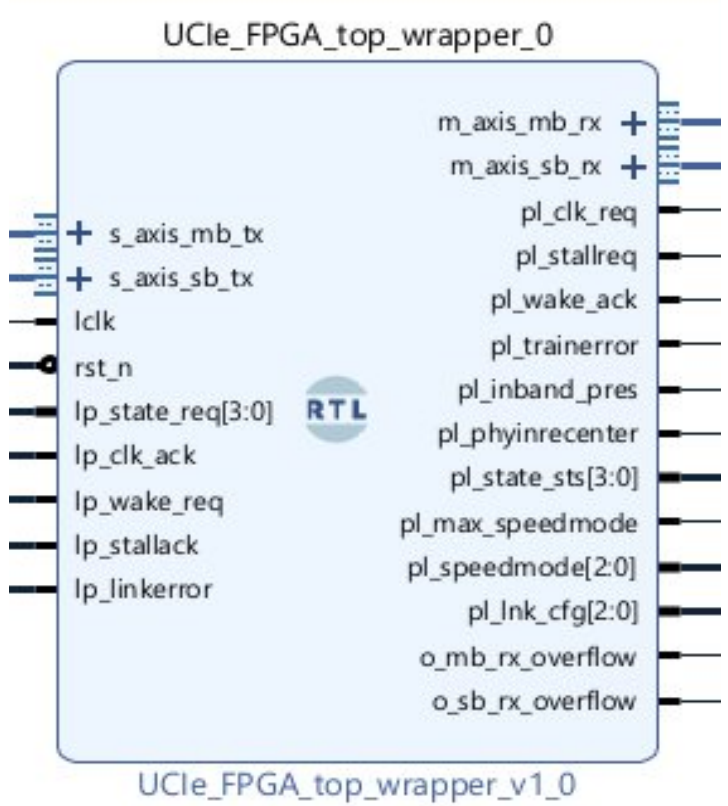
Hardware Validation



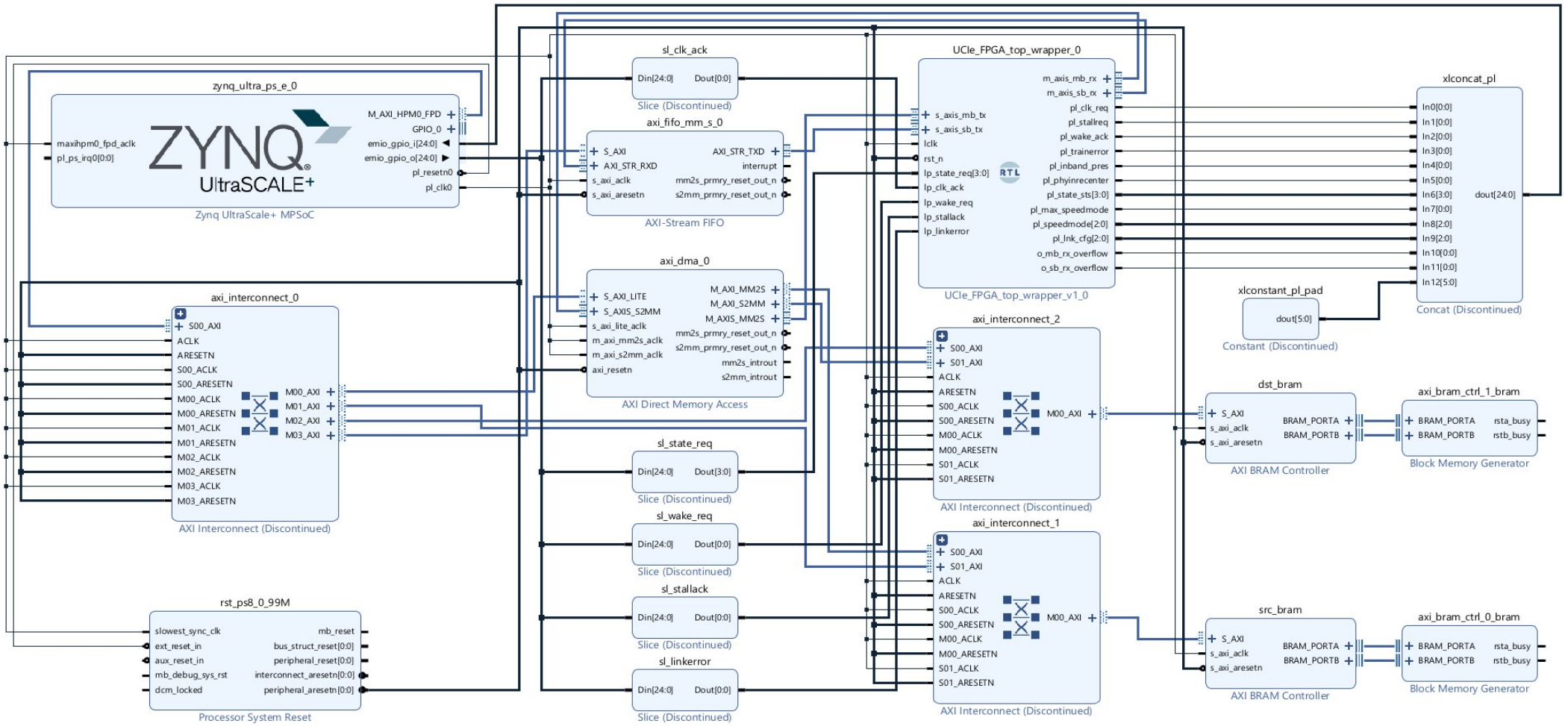
FPGA Prototype & Hardware Validation

- Packaged as a Vivado IP — PHY + both RDI/AXI bridges in one block
- AXI (HPM0): data & configuration — DMA, FIFO, BRAMs
- EMIO GPIO: RDI state control (lp_*) & link status (pl_*)*
- Controlled by a bare-metal C application on the Cortex-A53





FPGA Prototype & Hardware Validation



```
==================================================
   UCIe 3.0 PHY - FPGA Loopback Bring-up Software
==================================================
[1] Init DMA / Sideband FIFO / EMIO GPIO...
    OK.
[2] Programming PHY regs + starting training...
[DBG ---- WR  reg=0x001030 opc=0x09 data=0x00000000_00000000]
[DBG        1 ack-hi   ] stat=0x00000000 | clkreq=0 stall=0 wake=0 TERR=0 inband=0 recntr=0 state=0(RESET) sbov=0 mbov=0 || lp_state=9 clk_ack=1 stallack=0 | rxocc=2 txvac=508
[DBG        2 ack-lo   ] stat=0x00000000 | clkreq=0 stall=0 wake=0 TERR=0 inband=0 recntr=0 state=0(RESET) sbov=0 mbov=0 || lp_state=9 clk_ack=0 stallack=0 | rxocc=2 txvac=508
[DBG ---- WR cpl len=8 cpl1=0x80000000 st=0]
[DBG ---- WR  reg=0x001050 opc=0x09 data=0x00000000_003200A0]
[DBG        3 ack-hi   ] stat=0x00000000 | clkreq=0 stall=0 wake=0 TERR=0 inband=0 recntr=0 state=0(RESET) sbov=0 mbov=0 || lp_state=9 clk_ack=1 stallack=0 | rxocc=2 txvac=508
[DBG        4 ack-lo   ] stat=0x00000000 | clkreq=0 stall=0 wake=0 TERR=0 inband=0 recntr=0 state=0(RESET) sbov=0 mbov=0 || lp_state=9 clk_ack=0 stallack=0 | rxocc=2 txvac=508
[DBG ---- WR cpl len=8 cpl1=0x80000000 st=0]
[DBG ---- WR  reg=0x001004 opc=0x09 data=0x00000000_00200060]
[DBG        5 ack-hi   ] stat=0x00000000 | clkreq=0 stall=0 wake=0 TERR=0 inband=0 recntr=0 state=0(RESET) sbov=0 mbov=0 || lp_state=9 clk_ack=1 stallack=0 | rxocc=2 txvac=508
[DBG        6 ack-lo   ] stat=0x00000000 | clkreq=0 stall=0 wake=0 TERR=0 inband=0 recntr=0 state=0(RESET) sbov=0 mbov=0 || lp_state=9 clk_ack=0 stallack=0 | rxocc=2 txvac=508
[DBG ---- WR cpl len=8 cpl1=0x80000000 st=0]
[DBG ---- WR  reg=0x000010 opc=0x0D data=0x00000000_00000448]
[DBG        7 ack-hi   ] stat=0x00008020 | clkreq=0 stall=0 wake=0 TERR=0 inband=0 recntr=1 state=0(RESET) sbov=0 mbov=0 || lp_state=9 clk_ack=1 stallack=0 | rxocc=2 txvac=508
[DBG        8 ack-lo   ] stat=0x00008810 | clkreq=0 stall=0 wake=0 TERR=0 inband=1 recntr=0 state=0(RESET) sbov=0 mbov=0 || lp_state=9 clk_ack=0 stallack=0 | rxocc=2 txvac=508
[DBG ---- WR cpl len=8 cpl1=0x80000000 st=0]
    reg-write results: SETUP3(0x1030)=0 SETUP4(0x1050)=0 PHY_CTRL(0x1004)=0 LINK_CTRL(0x10)=0
    OK.
[3] Bringing RDI up to ACTIVE...
[DBG        8 inband   ] stat=0x00000011 | clkreq=1 stall=0 wake=0 TERR=0 inband=1 recntr=0 state=0(RESET) sbov=0 mbov=0 || lp_state=9 clk_ack=0 stallack=0 | rxocc=0 txvac=508
[DBG        8 req-ACT  ] stat=0x00008011 | clkreq=1 stall=0 wake=0 TERR=0 inband=1 recntr=0 state=0(RESET) sbov=0 mbov=0 || lp_state=1 clk_ack=0 stallack=0 | rxocc=0 txvac=508
[DBG        9 p2-wait  ] stat=0x00008850 | clkreq=0 stall=0 wake=0 TERR=0 inband=1 recntr=0 state=1(ACTIVE) sbov=0 mbov=0 || lp_state=1 clk_ack=1 stallack=0 | rxocc=0 txvac=508
[DBG        9 ACTIVE   ] stat=0x00008850 | clkreq=0 stall=0 wake=0 TERR=0 inband=1 recntr=0 state=1(ACTIVE) sbov=0 mbov=0 || lp_state=1 clk_ack=1 stallack=0 | rxocc=0 txvac=508
    SUCCESS: UCIe link is ACTIVE.
    Negotiated link parameters (sideband read of 0x14):
[DBG ---- RD  reg=0x000014 opc=0x0C]
[DBG       10 ack-hi   ] stat=0x00008850 | clkreq=0 stall=0 wake=0 TERR=0 inband=1 recntr=0 state=1(ACTIVE) sbov=0 mbov=0 || lp_state=1 clk_ack=1 stallack=0 | rxocc=4 txvac=508
[DBG       11 ack-lo   ] stat=0x00008850 | clkreq=0 stall=0 wake=0 TERR=0 inband=1 recntr=0 state=1(ACTIVE) sbov=0 mbov=0 || lp_state=1 clk_ack=0 stallack=0 | rxocc=4 txvac=508
    LINK_STATUS (0x14) = 0x00028900
      Link Up        : YES
      Negotiated wid : x16 (0x2)
      Negotiated spd : 8 GT/s (0x1)
      Retraining     : no
      Raw fmt / MP / EnhMP / x32AdvPkg : 0 / 0 / 0 / 0
[4] Sideband remote-message loopback...
[DBG       12 req^wait ] stat=0x00008850 | clkreq=0 stall=0 wake=0 TERR=0 inband=1 recntr=0 state=1(ACTIVE) sbov=0 mbov=0 || lp_state=1 clk_ack=0 stallack=0 | rxocc=0 txvac=508
[DBG   100011 req^TMO  ] stat=0x00008850 | clkreq=0 stall=0 wake=0 TERR=0 inband=1 recntr=0 state=1(ACTIVE) sbov=0 mbov=0 || lp_state=1 clk_ack=0 stallack=0 | rxocc=0 txvac=508
    WARN: remote message loopback failed (31) - skipping (not supported in this loopback). Continuing to MainBand test.
[5] MainBand DMA loopback (1024 bytes)...
==================================================
              TEST RESULT: PASS
==================================================
```

Conclusion & Future Work

Conclusion
Contributions & Limitations

✓ **Spec-compliant LTSM** — all normative states per UCIe 3.0.

✓ **FPGA prototype** — full digital PHY verified at 125 MHz, no timing violations.

✓ **Multi-level verification** — unit, wrapper, integration, and system level.

✗ **Excluded features** — PSPT, L2SPD, analog modeling, and multi-module out of scope.

✗ **Digital only** — PLL and AFE abstracted, no circuit-level modeling.

Future Work

Advanced Package — x64/x32 lane counts, per-lane repair, and redundant sideband support.

Multi-module support — arbitration layer above the current single-module wrapper.

PSPT & L2SPD — register hooks already in place, logic implementation pending.

ASIC synthesis — technology-specific place-and-route at UCIe target data rates.

UCIe-3D exploration — adapt digital PHY for direct die-to-die bonding variant.

Conclusion

Summary of work

Mainband Datapath: Full byte-to-lane mapping, scrambling, lane reversal, x16/x8/x4 configurations.

Sideband Subsystem: Complete packet engine with Sideband Feature Extensions and Performant Mode Operation.

Main State Machine: Full LTSM + RDI integration covering 10 normative states and 19 sub-states.

Register File: Complete UCIe register space with all 5 access attributes enforced at bit level.

Thank you